THESIS

STUDY OF A TERRAIN-BASED MOTION ESTIMATION MODEL TO PREDICT THE POSITION OF A MOVING TARGET TO ENHANCE WEAPON PROBABILITY OF KILL

by

Chin Beng Ang

September 2017

Thesis Advisor:	Morris Driels
Co-Advisor:	Isaac Kaminer

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2017	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE STUDY OF A TERRAIN-BASED MOTION ESTIMATION MODEL TO PREDICT THE POSITION OF A MOVING TARGET TO ENHANCE WEAPON PROBABILITY OF KILL	**5. FUNDING NUMBERS**
6. AUTHOR(S) Chin Beng Ang	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB number ____N/A____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

This thesis focuses on producing realistic time-space-position information (TSPI) data for moving targets as inputs for a weapon flyout model to compute the weapon Circular Error Probable (CEP) and the probability of damage. The velocity profile of the target is modeled based on the kinematic constraints for the type of vehicle and the type of path on which it is traveling. The discrete-time position profile can then be derived for the target and fed into the weapon flyout model to compute the associated measures of weapon effectiveness. The methodology is applied to both land vehicles traveling on specific roads, differentiated based on the road profiles, and small boats moving on the open sea to generate various sets of measures of weapon effectiveness for different scenarios. A program was also developed to enable a quick classification of the target's expected route of advancement into one of the six road types. This allows the user to perform quick analysis on the target's planned route of travel and produce measures of weapon effectiveness, which can then be readily utilized for swift decision making on the weapon options available.

14. SUBJECT TERMS velocity, position, estimation model, prediction, weapon effectiveness, curvature, road, boat, vehicle	**15. NUMBER OF PAGES** 129
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

STUDY OF A TERRAIN-BASED MOTION ESTIMATION MODEL TO PREDICT THE POSITION OF A MOVING TARGET TO ENHANCE WEAPON PROBABILITY OF KILL

Chin Beng Ang
Major, Republic of Singapore Air Force
B.S., National University of Singapore, 2006

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2017

Approved by: Morris Driels
Thesis Advisor

Isaac Kaminer
Co-Advisor

Garth V. Hobson
Chair, Department of Mechanical and Aerospace Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis focuses on producing realistic time-space-position information (TSPI) data for moving targets as inputs for a weapon flyout model to compute the weapon Circular Error Probable (CEP) and the probability of damage. The velocity profile of the target is modeled based on the kinematic constraints for the type of vehicle and the type of path on which it is traveling. The discrete-time position profile can then be derived for the target and fed into the weapon flyout model to compute the associated measures of weapon effectiveness. The methodology is applied to both land vehicles traveling on specific roads, differentiated based on the road profiles, and small boats moving on the open sea to generate various sets of measures of weapon effectiveness for different scenarios. A program was also developed to enable a quick classification of the target's expected route of advancement into one of the six road types. This allows the user to perform quick analysis on the target's planned route of travel and produce measures of weapon effectiveness, which can then be readily utilized for swift decision making on the weapon options available.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

CEP	Circular Error Probable
ECEF	Earth-Centered-Earth-Fixed
ENU	East-North-Up
GPS	Global Positioning System
IFS	In-flight Simulation
TSPI	Time-Space-Position Information
URL	Uniform Resource Locator
VBA	Visual Basic for Applications
WGS-84	World Geodetic System – 84

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

First, I wish to thank my advisors, Professor Morris Driels and Professor Isaac Kaminer, for their support and guidance throughout this project. Their wealth of knowledge and sophisticated approaches have been both very helpful and enlightening during the course of my thesis research.

In addition, I would like to express my gratitude to all my class instructors and professors who have helped me in one way or another during the course of my studies at NPS. They have provided me with an enriching and stimulating experience with their vast knowledge and continual encouragement during this long journey.

I would also like to thank my fellow schoolmates at NPS who showed a generous amount of hospitality to help my family to settle seamlessly into a new environment. The friendships forged during my stay here have certainly helped me while I have been far away from my home country.

Last, but certainly not least, I wish to express my gratitude toward my family for their utmost support during the course of my study, without which I would not have been able to stay the course.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

In modern warfare, with the rapid development of military technologies, armed forces around the world are increasingly using guided weapons to protect their valuable assets and to destroy targets of interest for strategic gains. Such weapons can be launched from various platforms, including aerial, surface, or even submerged ones. Even with the proliferation of the guided weapons, the ability of the weapons to hit moving targets remains a key area of interest. According to [1], the U.S. Navy is in the midst of equipping the submarine and the Tomahawk missile with enhanced seeker technology in order to better enable the weapon to take out mobile naval threats of the hostile forces. Meanwhile, [2] reveals that even the F-35, despite being the state-of-the-art fifth generation stealth fighter, is still looking for options to enhance its capability to hit moving targets.

Depending on the type of target environment, and the target's vulnerabilities, extensive weapon assessment is required to evaluate the likely weapon performance against such target, and the weapon delivery profile for the best possible outcome. To perform such assessment with models of higher fidelity, it is important to achieve a realistic representation of the target motion based on the target's type of environment.

This thesis focuses on the position estimation for both a fast-moving land vehicle along an expected route and a fast-moving boat on an open sea. The aim of the thesis is to develop a methodology to model an estimator for the likely path of travel for the land and sea targets of interest, and for the land target, to classify the target environment based on the road profile and vehicle characteristics; using this information enables a set of weapon performance estimates to be computed to facilitate quick decision making at the operational level. The goal of this thesis is to develop a realistic model characterizing the most probable paths of interest for specific target types based on the target environment to provide useful inputs that can be used to generate weapon performance metrics that enable the choice of weapon delivery for best possible outcome.

B. BACKGROUND

Weaponeering is defined in [3] as the process of "determining the quantity of a specific type of weapon required to achieve a defined level of target damage, considering target vulnerability, weapon effects, munitions delivery error, damage criteria, probability of kill, weapon reliability, etc." By determining the statistical estimate of the weapon's effectiveness against specific target, the user can then designate the appropriate type and quantity of weapons to be used to achieve the level of damage required. For weaponeering to achieve a close estimate of the expected weapon performance, it is important for the parameters assumed to match those used during weapon deployment in combat. Otherwise, an inaccurate model will lead to a poor weaponeering assessment, and result in inefficient or ineffective use of the weapons.

The area of interest is in making realistic weaponeering assessment against moving targets. Broadly, this can be classified into two scenario types, constrained and unconstrained. The first scenario type involves targets moving along a defined path, for example, a vehicle going down a particular stretch of road or a train on a railroad track, while the latter scenario involves targets moving in an open environment, for example, a tank traversing a desert terrain or a speedboat traveling in an open sea.

C. THESIS OVERVIEW

For this thesis, the focus is on both scenario type, particularly weaponeering assessment against specific type of land vehicles moving along a certain stretch of road and small fast-moving boats in the open sea. An overview of the weapon assessment process is illustrated in Figure 1.

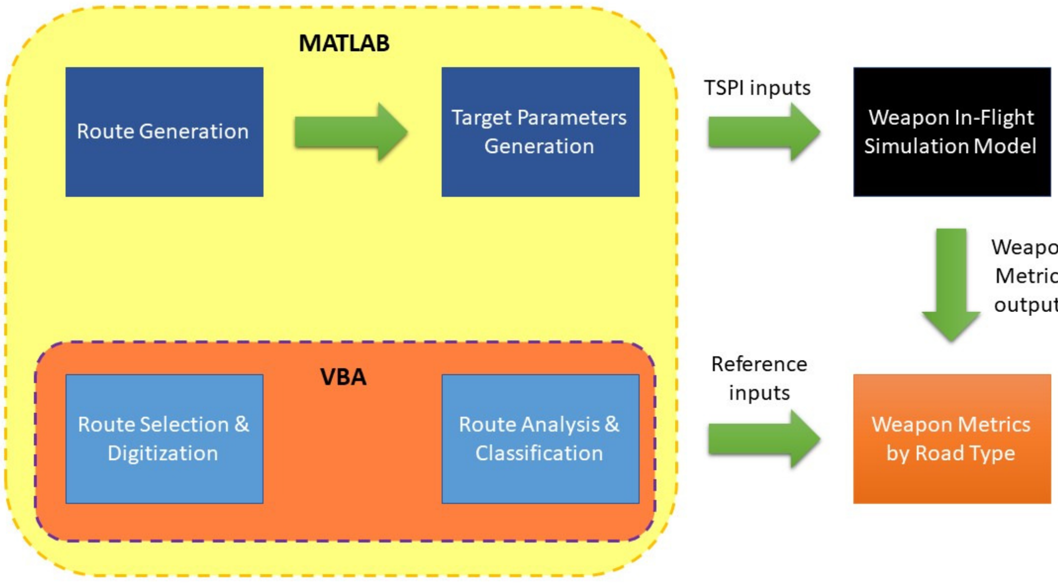

MATLAB
Route Generation
Target Parameters Generation
TSPI inputs
Weapon In-Flight Simulation Model
Weapon Metrics outputs
VBA
Route Selection & Digitization
Route Analysis & Classification
Reference inputs
Weapon Metrics by Road Type

Figure 1. Overview of Weapon Assessment Process Flow

Ideally, to perform a realistic weapon assessment, the ability to precisely predict the exact position of the target of specific vehicle type will enable the most accurate simulation of the actual combat scenario, and thereby generate the weapon metrics to best represent the likely outcome with the highest fidelity. Nevertheless, given the large amount of time required for the weapon In-Flight Simulation (IFS) model to produce the simulation results for each and every type of weapon for a particular mission, it would be impractical to have this carried out during the actual mission phase when time is limited.

Therefore, for the case of the land target, it is necessary to categorize the wide distribution of roads into discrete road types from which a random road, which is representative of each of the road types, is used in the weapon IFS model for the full weapon assessment to be carried out. Similarly, for the case of the sea target in an unconstrained environment, it is necessary to predict the likely evasive maneuvers that the boat will adopt in its approach, and carry out the full weapon assessment in the weapon IFS model. This forms the first part of the thesis.

Due to the vast number of roads available, it is nearly impossible to carry out the full weapon assessment whenever a new target emerges. As such, a generalization of the types of roads available is required, and a representative road of each type is to be generated and analyzed. In Chapter II a discussion of the generation of synthesized roads of different types based on type of road profile expected for each type is offered for the land target. The discussion addresses how the various roads are categorized and how the representative road for each category is generated. For the sea target, the route that a boat is likely to follow in its attempt to evade weapons during its escape or attack is designed and generated using an existing realistic boat simulation model.

In Chapter III, we examine how the velocity profile of the land vehicle traveling along each of the synthesized roads generated can be computed based on the road profile. It is important to model the position and velocity profile of the target vehicle as closely as possible to its actual motion to ensure that the simulation results from the IFS model will be nearly identical to the actual weapon performance under actual combat scenario. For the sea target, the velocity profile is generated as part of the simulation. Further processing of the data generated is required, however, to transform the data points into the Time Space and Position Information (TSPI) format necessary for the weapon IFS model.

Thereafter, Chapter IV contains a study of how the position of both the land and sea vehicles, based on the velocity profile computed in Chapter III, can be determined and generated at discrete time intervals, so that the TSPI can be fed into the weapon IFS model for the weapon simulations to be carried out. The IFS model requires the TSPI inputs for the target to be at one-second intervals, along with the velocity profile during each time-step.

The weapon IFS model simulations will not be part of this thesis due to the classified database from which it draws the weapon information. The weaponeering community will make use of the TSPI inputs to generate the associated simulation results. Similarly, the simulation results are classified and will not be disclosed in this thesis.

The second part of the thesis then focuses on enabling the users to make use of simulation results generated earlier for making weaponeering assessment during the actual mission planning. For this part of the thesis, the focus is on the land target scenario, due to the unconstrained nature of the sea target scenario.

Chapter V presents the digitization process of an actual route of interest that a target is likely to travel on. It discusses how the map selection process can be carried out, and how the route information can be imported into a program and processed into a digital form that can be used for further target velocity profile analysis.

Chapter VI then makes use of the digitized route to compute the likely velocity profile of the target, using such information to determine the route profile and classify the route of interest into one of the road type classification discussed in Chapter II. By classifying the route into one of the road types, the associated weapon metrics for the particular road type can then be referred to for a close estimate of the weapon metrics obtained from the elaborate weapon simulations already carried out for the specific road type.

Chapter VII summarizes the conclusions from this study and offers recommendations for further studies.

D. LITERATURE REVIEW

In this section, we discuss the results of the review carried out on existing literature for the following areas to aid the study.

1. Road Profile Derivation

To compute a realistic velocity estimation for a vehicle traveling down a particular stretch of road, it is necessary to relate the maximum design speed limit with the road curvature along various stretches of the road. And to do that, it is necessary to compute the curvature at various points along the entire route.

Andersson and Aronsson [4] discuss the road shape modeling concept and the rationale for the selection of the spline interpolation of the points. The splines constructed are approximations, however, and the splines may not completely follow the exact path of interest despite the degree of shaping done. Hence, an alternate method of determining the road profile is required.

With the digitization of the route, the route will be well defined by many points along the entire stretch of the path. Thus, an exact method of finding the curvature at each and every point along the path can be used. By evaluating the circle that passes through the point and its immediate two adjacent points, it is possible to determine the curvature of the road. This will be discussed further in Chapter III.

Once the curvature of the path has been obtained, the speed limit at each point along the road can then be computed. Donnell et al. [5] discuss the relationship between the horizontal road curvature and the superelevation. By considering the centripetal acceleration of the vehicle when it travels along a curved road, as shown in Figure 2, the velocity of the vehicle, as a function of the road curvature, the side friction factor and the road superelevation, can be determined.

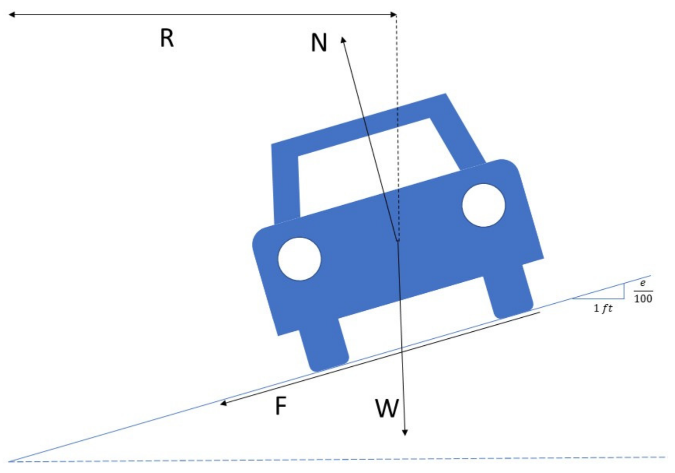


Figure 2. Centripetal Acceleration of Vehicle Traveling along Curved Road. Adapted from [5].

Resolving forces to both the horizontal and vertical directions, and equating forces in the horizontal direction only, we have

$$F\cos\theta + N\sin\theta = \frac{mV^2}{R}. \tag{1.1}$$

Similarly, for the vertical direction,

$$N\cos\theta - W - F\sin\theta = 0. \tag{1.2}$$

The relation between F and R is given by

$$F = \mu N. \tag{1.3}$$

Using Equation (1.3), and equating R in both (1.1) and (1.2),

$$\frac{W}{\cos\theta - \mu\sin\theta} = \frac{mV^2}{R(\mu\cos\theta + \sin\theta)}. \tag{1.4}$$

Solving for V,

$$V^2 = \frac{gR(\mu + \tan\theta)}{1 - \mu\tan\theta} \cong gR(\mu + \tan\theta). \tag{1.5}$$

With Equation (1.5), the following equation is obtained:

$$\frac{V^2}{15R} = 0.01e + f, \tag{1.6}$$

where

e = rate of superelevation, in percent

f = side friction (demand factor)

V = vehicle speed, in mph

R = radius of curve, in feet

With Equation (1.6), the design speed, limited by the road curvature, can thus be determined and incorporated as one of the constraints on the vehicle in subsequent velocity profile analysis.

2. High-Speed Boat Maneuver Estimation

In order to more accurately predict the evasive maneuvers that the boat will adopt in its approach, it will be essential to first determine how maneuverable the vessel is. The maneuverability will ultimately determine the type of turns the vessel is capable of, as well as the time required and associated speed attainable with such turns.

Aslan [6] discusses the maneuvering characteristics of some of the fast craft employed in different parts of the world. Incorporating the correlation derived by Lewandowski [7] between the fast craft turning radius with the length LWL, volume ∇, volumetric Froude number F_∇ and rudder angle δ, based on the empirical data from the works of Denny and Hubble [8], the following equation was developed.

$$\frac{R_c}{LWL} \approx 0.5\left[1.7 + 0.0222 F_\nabla \left(\frac{LWL}{\nabla^{\frac{1}{3}}}\right)^{2.85}\right]\left(\frac{30}{\delta}\right). \tag{1.7}$$

The Froude number, in turn, is derived from the vessel approach speed U_a and the volume ∇ through the following equation.

$$F_\nabla = \sqrt{\frac{U_a}{g\nabla^{\frac{1}{3}}}}. \tag{1.8}$$

The normalized turning radius $\frac{R_c}{LWL}$ computed and the approach speed U_a can then be used to derive the speed during the turn U_c through the following equation by Denny and Hubble [8].

$$U_c = U_a \sqrt{\frac{1}{1 + K_c\left(\frac{LWL}{R_c}\right)^2}}. \tag{1.9}$$

The empirical constant K_c can be obtained using the Froude number F_∇ and rudder angle δ using the following equation.

$$K_c = \frac{30F_\nabla{}^2}{\delta}. \tag{1.10}$$

To ensure the stability of the vessel during a turn, the Nomoto K-T model is studied in [6]. To simplify the model, the first order transfer function by Nomoto was adopted. When expressed in time domain, the following equation is derived and used in the model.

$$T\dot{r} + r = K\delta. \tag{1.11}$$

where

$$r = \dot{\psi}. \tag{1.12}$$

Assuming the yaw inertia coefficient T to be zero, Aslan [6] then derives the following relation between the turning ability K and the vessel approach speed U_a, the turning radius R_c and the max rudder angle δ_{max}.

$$K = \frac{U_a}{R_c \times \delta_{\max}}. \tag{1.13}$$

Using the computed value of K, substituting Equation (1.13) into Equation (1.11) and introducing a correction factor η of value 0.909 to ensure consistency with the turning radius R_c in the Lewandowski's equation [7], the following equation is obtained.

$$\dot{\psi} = K\delta\eta. \tag{1.14}$$

Equation (1.14) can then be integrated with respect to time for the computation of the velocity profile using the following equations.

$$\begin{aligned} \dot{x} &= U_c \sin\psi \\ \dot{y} &= U_c \cos\psi \end{aligned}. \tag{1.15}$$

Further integration of Equation (1.15) can generate the position profile of the vessel with respect to time.

3. Map Coordinates Transformation

To digitize a road, it is necessary to convert map locations in the Geodetic Coordinates System to that of a local reference frame. Both Farrell [9] and Driels [3] discuss the conversion of the Global Positioning System (GPS) readings from the WGS-84 geodetic system to a local East-North-Up (ENU) Cartesian axis.

The basic reference frame for terrestrial navigation is the Earth-Centered-Earth-Fixed (ECEF) geodetic coordinate system, where the coordinates are in latitude ϕ, longitude λ, and altitude h. This is also referred to as the global reference frame, which is most commonly associated with the GPS.

An alternate reference frame is known as the ECEF rectangular system, where the x-axis passes from the origin at the Earth's center to the prime meridian, the z-axis passes through the North Pole, and the y-axis orthogonal to these two axes. This frame is convenient, as it allows for easy relation to the ENU frame that is used subsequently.

To use the ECEF geodetic system, the shape of the Earth's ellipsoid needs to be defined, and the parameters for the assumed shape used in GPS, known as the World Geodetic System – 1984 (WGS-84), are defined by the equatorial and polar radii, a and b, of 6,3778,137.0 m and 6,356752.3142 m, respectively. The ellipse is presented in Figure 3.

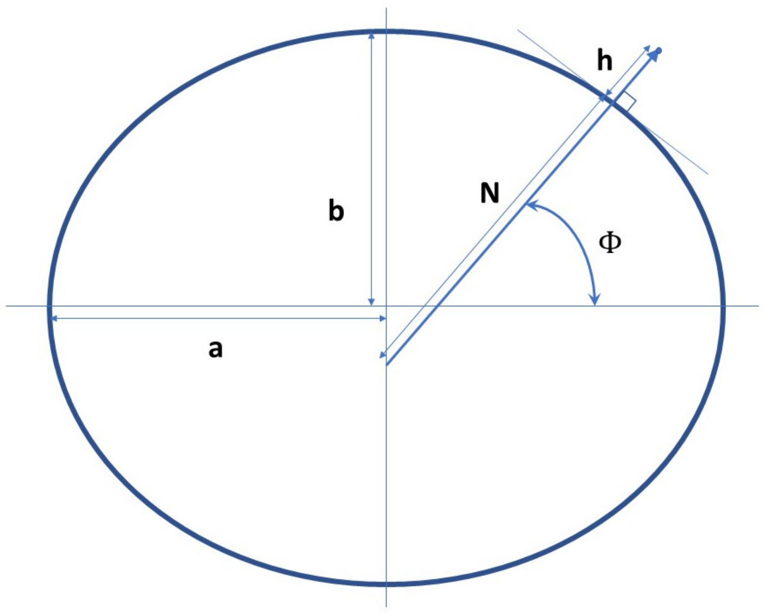


Figure 3. WGS-84 Ellipse

The ellipsoid flatness is defined by

$$f = \frac{a-b}{a}. \tag{1.16}$$

The eccentricity of the ellipsoid is defined by

$$e = \sqrt{f\left(2-f\right)}. \tag{1.17}$$

The distance from the surface of the ellipsoid to the z-axis along the ellipsoid normal, as a function of ϕ, is defined by

$$N(\phi) = \frac{a}{\sqrt{1-e^2\sin^2\phi}}. \tag{1.18}$$

The ECEF rectangular coordinates can then be calculated from the geodetic system using the following:

$$x = \left(h+N\right)\cos\phi\cos\lambda, \tag{1.19}$$

$$y = \left(h+N\right)\cos\phi\sin\lambda, \tag{1.20}$$

$$z = \left(h + \left(1 - e^2\right) N\right) \sin \phi . \tag{1.21}$$

Thereafter, in order to form a right-handed coordinate system that is analogous to the Cartesian coordinate system, it is essential to further transform from the ECEF rectangular system to that of the ENU system. This rectangular system is defined with the origin at a point of interest on the Earth's surface, with the vertical axis along the ellipsoid normal, and the other two axes orthogonal and aligned to the local geographic East and North directions. The ENU is presented in Figure 4.

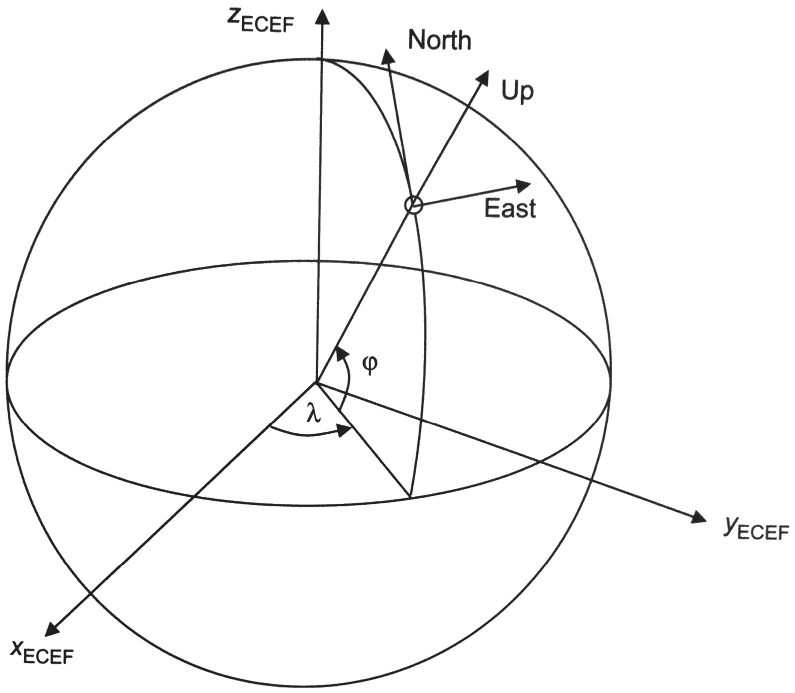


Figure 4. ECEF Geodetic and ENU Coordinate Frame. Source: [3].

With the origin of the ENU defined as (x_0, y_0, z_0) in the ECEF rectangular frame, to transform the ECEF rectangular frame to the ENU frame would require a translation of the origin to (x_0, y_0, z_0), followed by a rotation of the axes to align with those of the ENU. The complete transformation is given by

$$\begin{pmatrix} e \\ n \\ u \end{pmatrix} = \begin{bmatrix} -\sin\lambda & \cos\lambda & 0 \\ -\sin\phi\cos\lambda & -\sin\phi\sin\lambda & \cos\phi \\ \cos\phi\cos\lambda & \cos\phi\sin\lambda & \sin\phi \end{bmatrix} \begin{bmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{bmatrix}. \tag{1.22}$$

With the transformation in Equation (1.22) into the ENU system, which is analogous to the familiar Cartesian system, the digitized route information can then be easily further processed in subsequent analysis.

THIS PAGE INTENTIONALLY LEFT BLANK

II. ROUTE GENERATION

A. LAND ROUTE GENERATION

In this section, the aim is to generate routes that are representative of the different road types. Before we begin to generate synthesized road for each road type, it is essential to first define a quantitative approach to classify the road types. Thereafter, we will generate the roads based on the characteristic parameter associated with each road type.

1. Statistical Road Type Classification

Driels [10] introduced the concept of road type characterization through the analysis of actual roads. A statistical analysis is carried out to study both the heading and change in heading for four distinct types of real roads to determine their respective means and standard deviation. The statistical results, in histogram form, are presented in Figures 5 to 8.

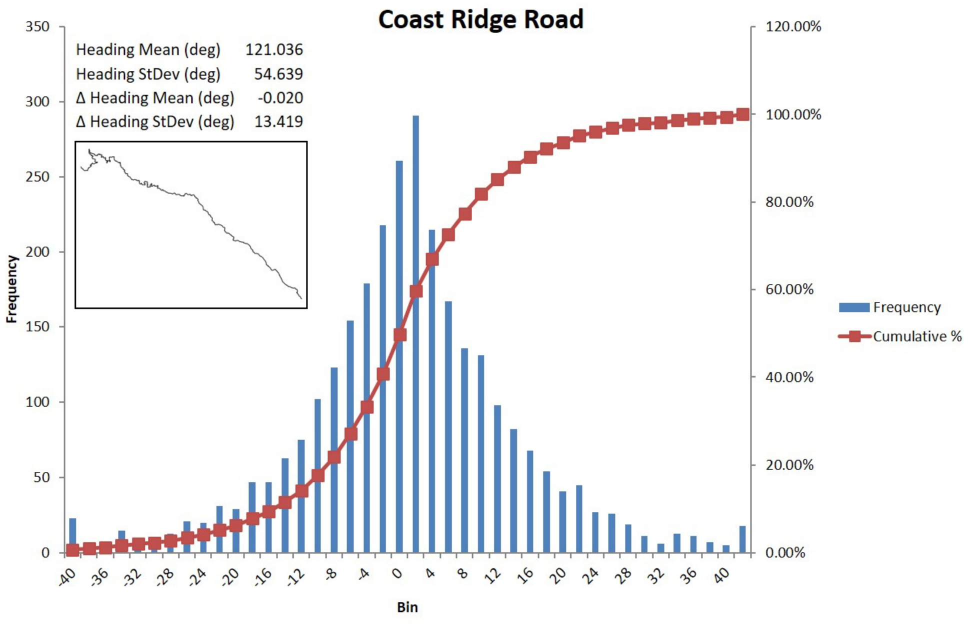


Figure 5. Statistical Analysis of Coast Ridge Road. Source: [10].

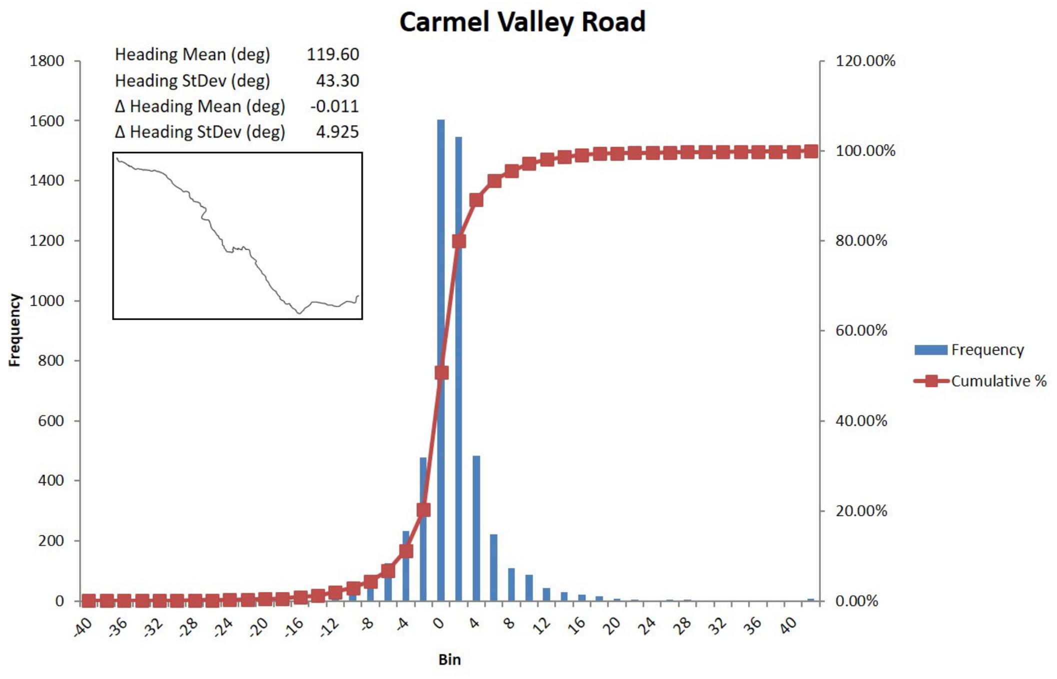


Figure 6. Statistical Analysis of Carmel Valley Road. Source: [10].

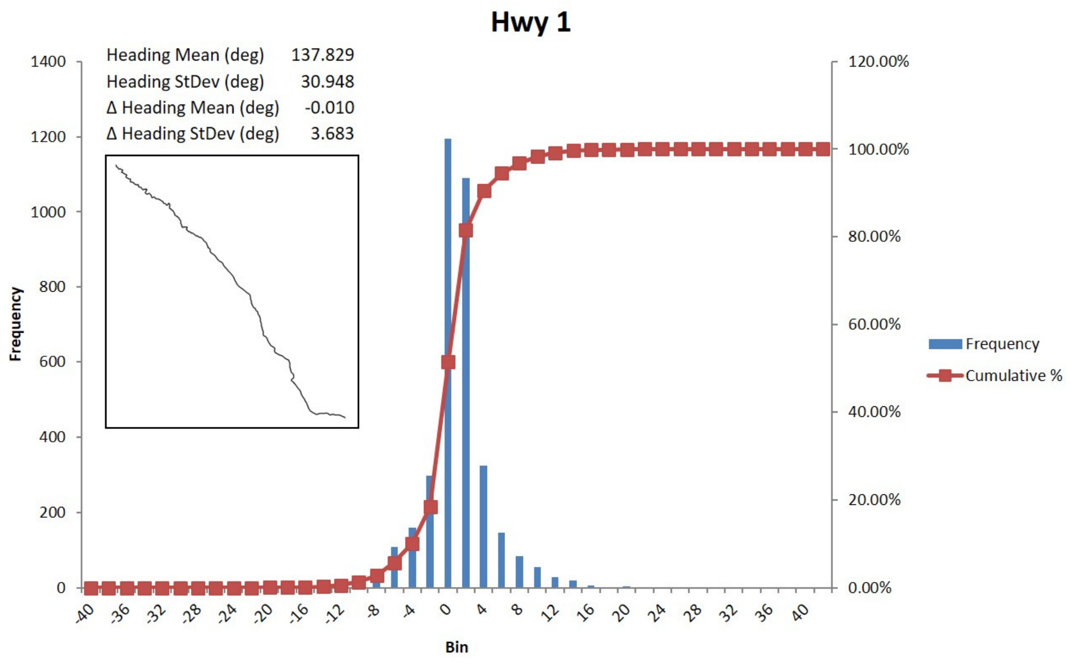


Figure 7. Statistical Analysis of Highway 1. Source: [10].

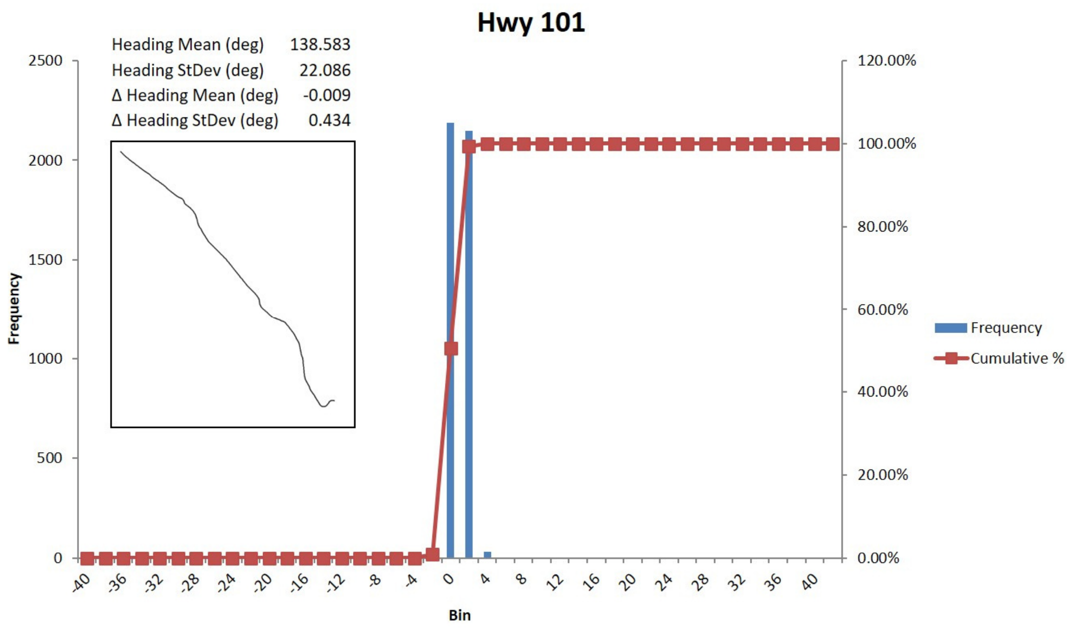


Figure 8. Statistical Analysis of Highway 101. Source: [10].

The results of the preceding statistical analyses are consolidated in Table 1 for ease of comparison.

Table 1. Statistical Analyses of Actual Roads

Road type	Mean heading	Sigma heading	Mean change in heading	Sigma change in heading
Coast Ridge Road	121	54.6	-0.02	13.42
Carmel Valley Road	120	43.3	-0.01	4.93
Pacific Coast Hwy (Hwy 1)	138	31.0	-0.10	3.63
Highway 101	139	21.1	-0.01	0.49

From the results obtained, it can be easily observed that the change in road heading generally follows a normal distribution, and the standard deviation in the change of heading increases with the number of curves and degree of curvatures expected on the roads. From this, for the synthesized road generation, the synthesized roads are assumed to follow a normal distribution and parameterized by the standard deviation of the road change in heading. A total of six different road types are identified, and their associated standard deviations in change of heading are summarized in Table 2.

Table 2. Road Type Classification by Change in Road Heading

Road type	Sigma change in heading	Mean sigma change in heading
1	0< to ≤1	0.5
2	1< to ≤4.5	2.5
3	4.5< to ≤9.5	7
4	9.5< to ≤14.5	12
5	14.5< to ≤19.5	17
6	19.5< to ≤30.5	25

2. Route Synthesis by Road Type

Using the parameters defined in Table 2 alone is not enough to fully characterize and produce synthesized roads that are representative of the original road. A comparison of the original Carmel Valley and the synthesized road reproduced using the same road parameters is presented in Figure 9.

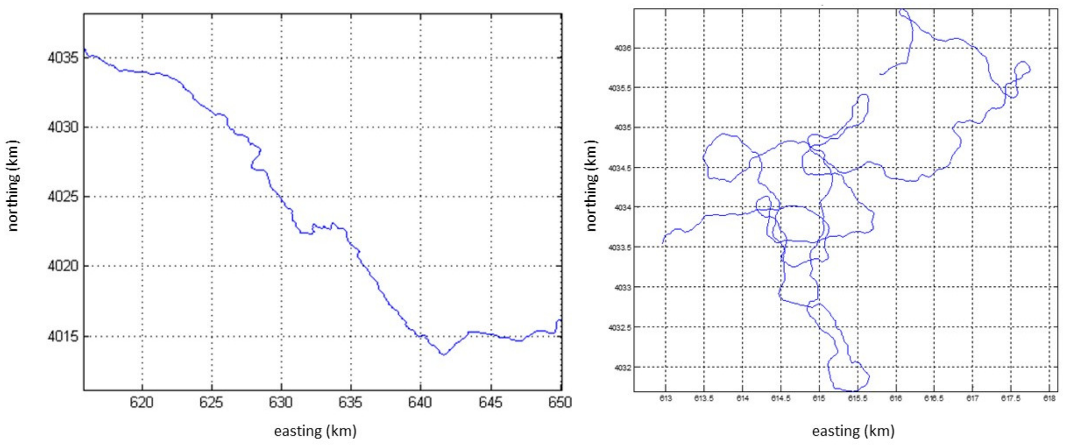


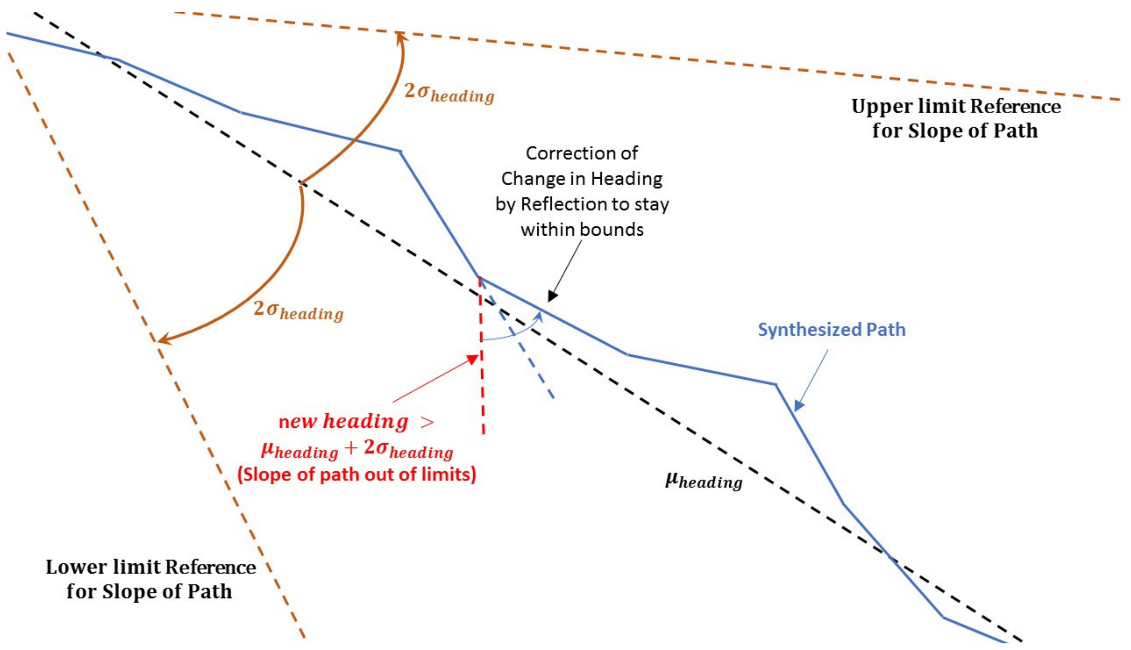
Figure 9. Comparison of Carmel Valley Road and Synthesized Reproduction

It can be clearly seen that the synthesized reproduction of the Carmel Valley road is totally random and lacks a general direction of travel. Therefore, synthesizing roads in such an unconstrained manner is not suitable, and additional constraints are required to produce synthesized roads that can be more representative of the original. To prevent excessive randomness in the path generation and to impose a general road heading, the mean heading and the standard deviation in the heading are included for the road synthesis.

The road synthesis is carried out by iteratively generating small segments of path lengths with change of heading randomly drawn from a normal distribution with the mean and parameters as defined in Table 3. These road segments are then incrementally added on to an arbitrarily chosen starting point until a finite route distance is achieved. In addition to this, a further constraint is also imposed on the road heading by restricting the maximum change in heading to no more than two standard deviations of the heading. This is illustrated in Figure 10.

Table 3. Final Road Type Classification

Road type	Mean heading	Sigma heading	Sigma change in heading	Mean sigma change in heading
1	135	25	0< to ≤1	0.5
2	135	29	1< to ≤4.5	2.5
3	135	41	4.5< to ≤9.5	7
4	135	53	9.5< to ≤14.5	12
5	135	66	14.5< to ≤19.5	17
6	135	73	19.5< to ≤30.5	25



$2\sigma_{heading}$
Upper limit Reference for Slope of Path
Correction of Change in Heading by Reflection to stay within bounds
$2\sigma_{heading}$
Synthesized Path
new heading $> \mu_{heading} + 2\sigma_{heading}$ (Slope of path out of limits)
$\mu_{heading}$
Lower limit Reference for Slope of Path

Figure 10. Route Synthesis Path Constraint

With these additional constraints imbued into the program, a new synthesized road is generated for verification of the updated program. The new synthesized road, while not exactly the same as the actual Carmel Valley road, which is expected due to the randomness in the generation of paths during the road synthesis, is much closer to the actual road than before, and more representative of a road in the specific road type. The roads are provided in Figure 11 for comparison.

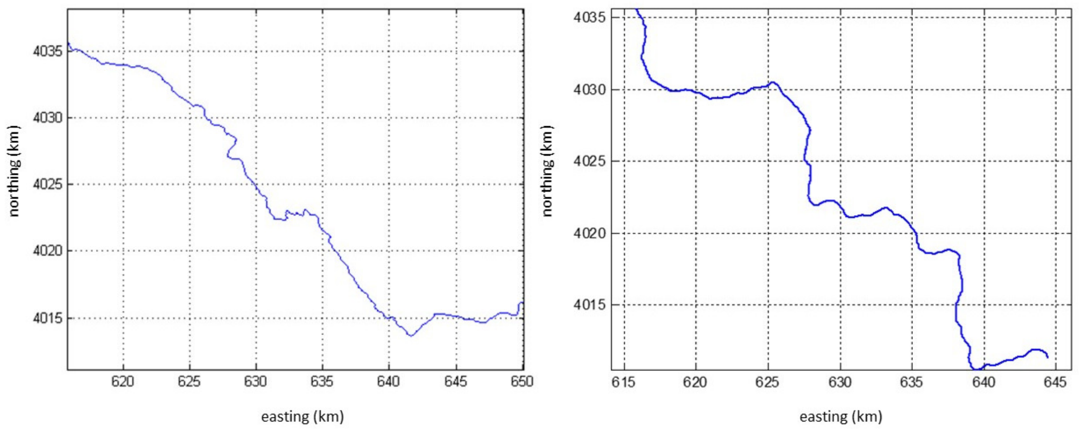


Figure 11. Comparison of Digitized Carmel Valley Road with New Synthesized Road

With the new changes incorporated, and having verified the ability of the program to produce a synthesized road that is reasonably representative of actual digitized road, the representative synthesized roads for each of the six distinct road types are generated and presented in Figures 12 to 17. The Cartesian coordinates of all the points making up the newly synthesized roads are saved for subsequent work in Chapter III for conversion into the TSPI input format required for weapon performance analysis by the IFS model.

As intended, the roads generated all have a common heading towards the southwestern direction, and the road profile gets increasingly winding as the standard deviation in the change of heading increases. From the plots of the routes generated, it can be easily observed that the routes are as per design, and they are ready to be worked on further in subsequent chapters. A copy of the MATLAB codes for the generation of routes for the various road types can be found in Appendix A.

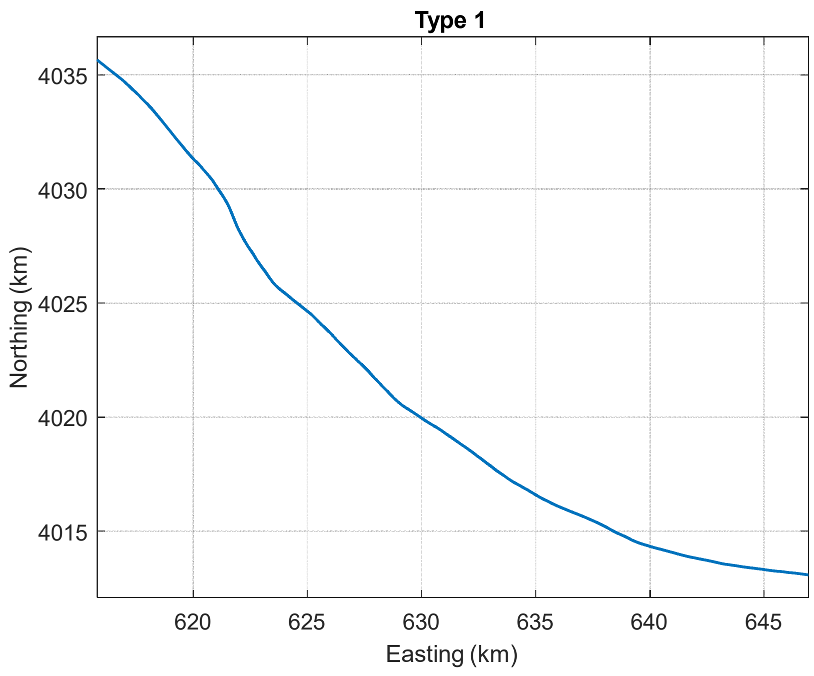


Figure 12. Synthesized Road for Type 1

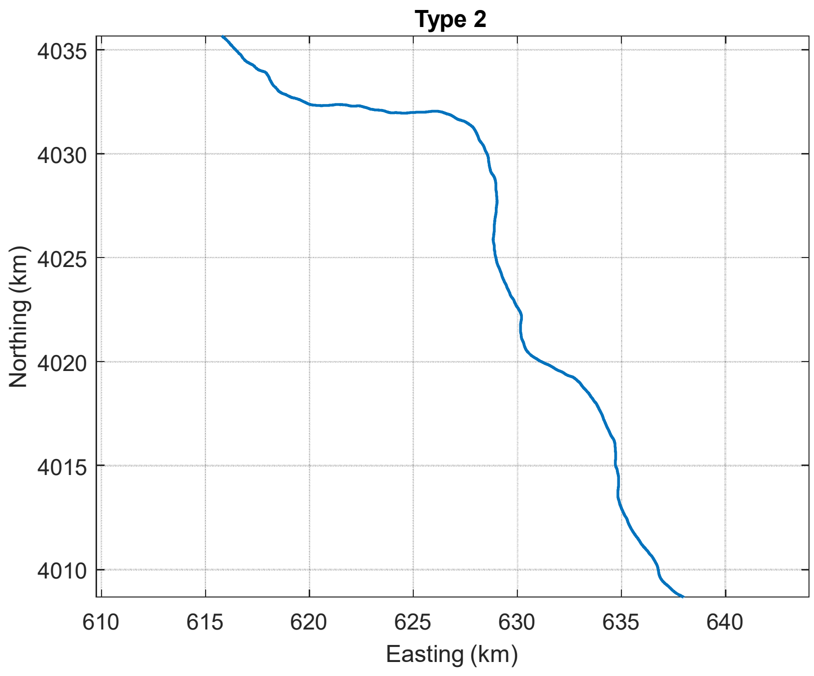


Figure 13. Synthesized Road for Type 2

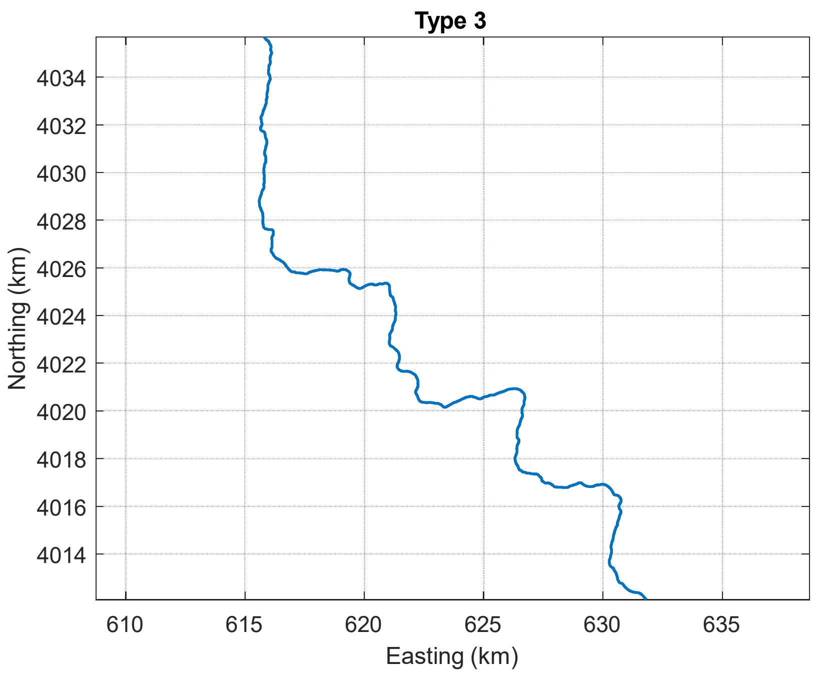


Figure 14. Synthesized Road for Type 3

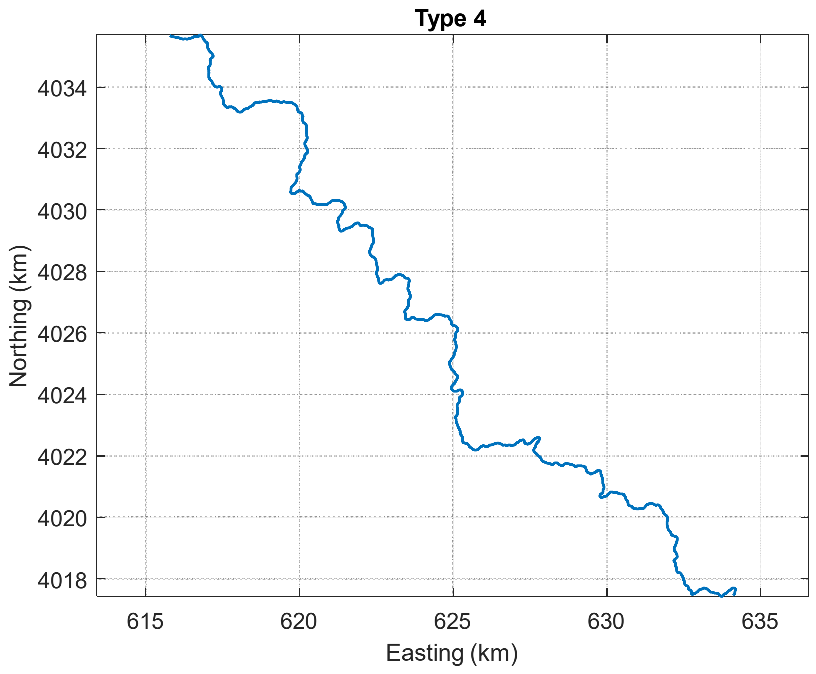


Figure 15. Synthesized Road for Type 4

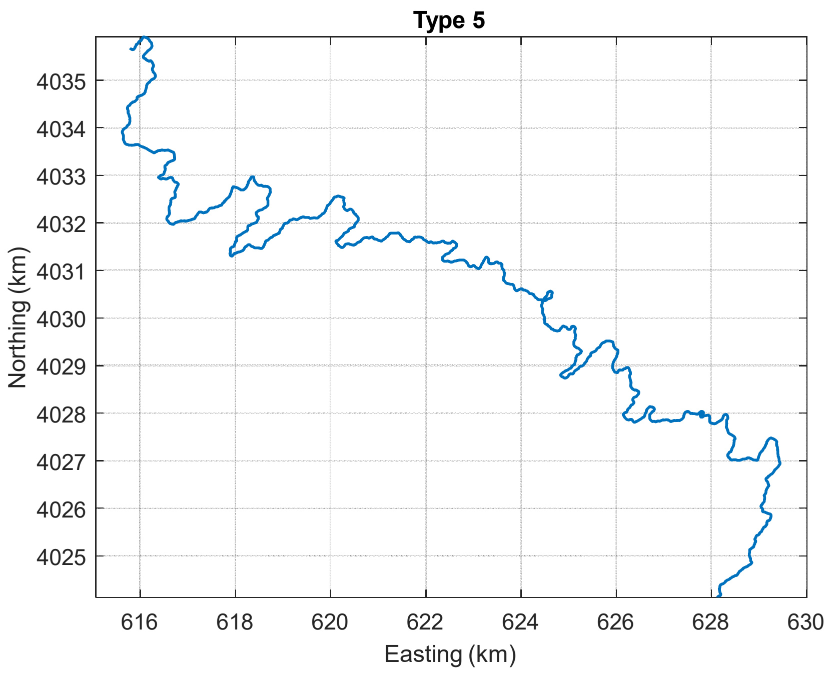


Figure 16. Synthesized Road for Type 5

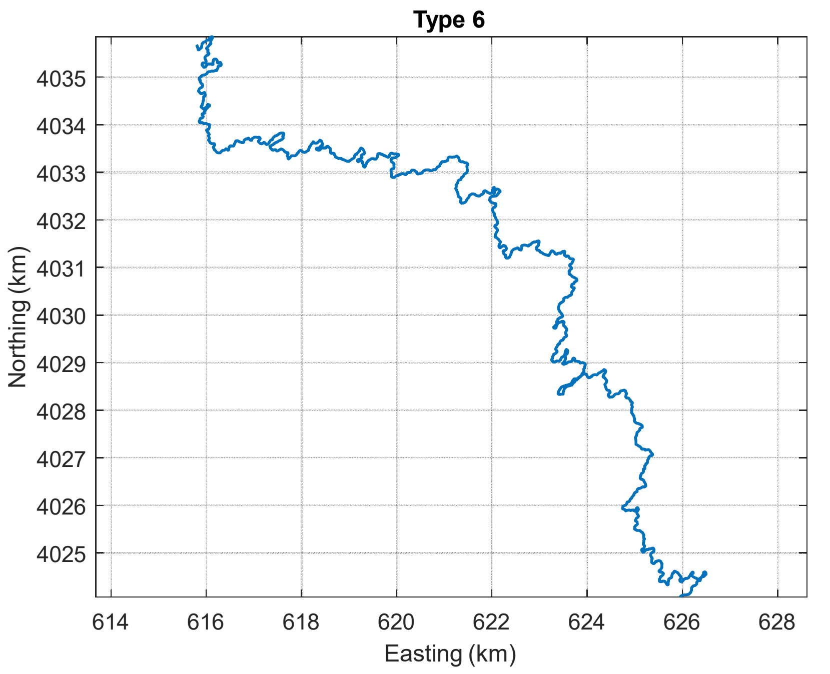


Figure 17. Synthesized Road for Type 6

B. SEA ROUTE GENERATION

In earlier discussion, the target's route of travel is constrained by a specific path, and these paths can in turn be characterized by their profile into different road types, from which a representative path can be generated for further analysis. Yet, for a vessel moving in the open sea, there is no such constraint to restrict it within a fixed path. Instead, it is free to move wherever it wants, whenever it wants. In order to generate a realistic prediction of the target's path of travel on the open sea, certain assumptions about its travel are required.

For this study, it is assumed that the general heading for the target, when viewed on an X-Y Cartesian plane, is towards the right, along the x-axis. To reach its target in the shortest amount of time, it always maintains its approach speed at the maximum possible. To try to evade a weapon directed at it, though, the vessel must execute turns at the highest turning speed possible, while maintaining its general motion towards the desired heading. This is termed as the desired travel path for the sea vessel, as illustrated in Figure 18.

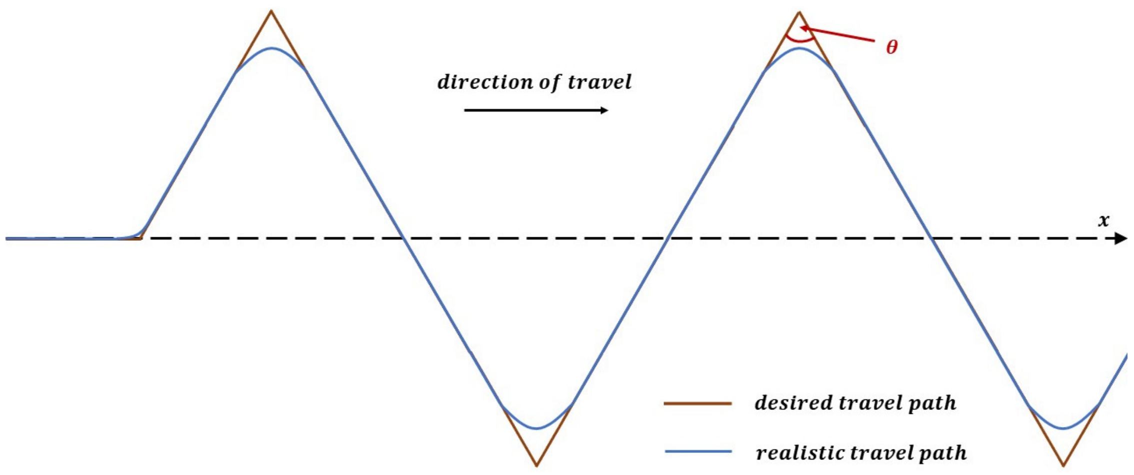


Figure 18. Sea Vessel Predicted Path of Travel with Evasive Maneuvers

Of course, a vessel cannot perform sharp turns, as depicted by the desired travel path in Figure 18, due to the turning radius required when making turns at high speeds. As such, a more realistic path for a sea vessel executing evasive maneuvers at high speeds is more accurately represented by the blue path, as shown in Figure 18.

Furthermore, for the turning angle θ, the sharper the angle, the higher is the speed of the sea vessel along the x-direction. Thus, a large angle will enable the target to move down the x-axis in the shortest possible time while executing the evasive maneuvers. A large angle turn, however, will not be aggressive enough for an evasive maneuver. Hence a sharp turn angle of 60 degrees is used in the predicted route of the sea vessel to simulate more aggressive evasive maneuver while maintaining a substantial speed along the x-axis for the purpose of this study.

III. VELOCITY PROFILE COMPUTATION

A. VELOCITY PROFILE FOR LAND VEHICLE

In this section, the aim is to determine the velocity profile of the land vehicle traveling along a constrained path. The model maximum speed of the vehicle is constrained by both the vehicle maximum speed and the road design speed. As mentioned in Chapter I, in order to make use of Equation (1.6) to compute the maximum road design speed limit along a curve, it is necessary to determine the radius of curvature.

1. Radius of Curvature Computation

Six distinct routes, made up of thousands of intermediate points with their associated Cartesian coordinates, have been generated earlier. Due to the large number of points in each route, it is assessed that it will be computationally less intensive to solve for the radius of curvature at a point using a circle that passes through the point and the two points immediately adjacent to it, as shown in Figure 19. Furthermore, this method produces an exact solution for the radius of curvature, while there may be certain loss of accuracy, even with good shaping, with the method of spline interpolation, as suggested by Andersson [4].

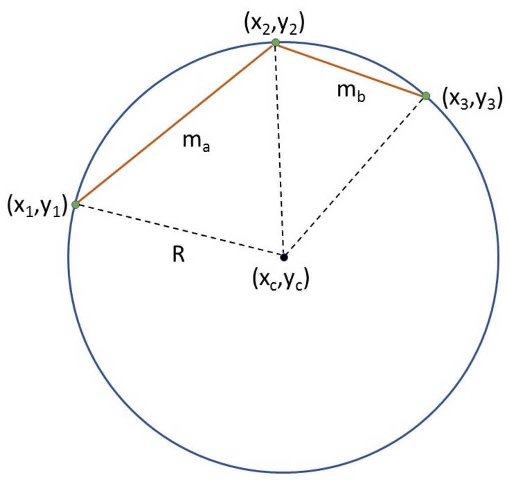


Figure 19. Radius of Curvature Computation Using Three Known Points

The gradients of the lines connecting the first and second pair of points are given by

$$m_a = \frac{y_2 - y_1}{x_2 - x_1}, m_b = \frac{y_3 - y_2}{x_3 - x_2}. \tag{3.1}$$

The equation of the lines connecting the midpoint of line of gradient m_a and that of gradient m_b to the center of the circle of curvature are given by

$$y_c - \frac{y_1 + y_2}{2} = -\frac{1}{m_a}\left(x_c - \frac{x_1 + x_2}{2}\right), y_c - \frac{y_2 + y_3}{2} = -\frac{1}{m_b}\left(x_c - \frac{x_2 + x_3}{2}\right). \tag{3.2}$$

Equating the value of y_c using the equation of lines in (3.2), we get

$$\begin{aligned} \frac{y_1 + y_2}{2} - \frac{1}{m_a}\left(x_c - \frac{x_1 + x_2}{2}\right) &= \frac{y_2 + y_3}{2} - \frac{1}{m_b}\left(x_c - \frac{x_2 + x_3}{2}\right) \\ x_c\left(\frac{1}{m_b} - \frac{1}{m_a}\right) &= \frac{y_3 - y_1}{2} - \frac{x_1 + x_2}{2m_a} + \frac{x_2 + x_3}{2m_b} \\ x_c &= \frac{m_a m_b (y_3 - y_1) - m_b (x_1 + x_2) + m_a (x_2 + x_3)}{2(m_a - m_b)} \end{aligned} . \tag{3.3}$$

Using Equations (3.2) and (3.3), the center of the circle of curvature, and hence the radius of curvature, can be computed for every point on the synthesized route, less the start and end points. Having determined the radius of curvature, we can then commence on establishing the velocity profile of a target traveling along the designated route.

2. Maximum Model Speed Limit Computation

Having computed the radius of curvature in the previous section, we can then proceed to compute the maximum road design speed at each point of the route based on the road profile using the Equation (1.6) discussed by Donnell [5], as mentioned in Chapter II. Ideally, the vehicle should travel at the road design speed limit in order to minimize the time taken to reach the destination. However, this is only possible if either the route that the vehicle is traveling on is fairly straight with little change on the maximum speed limit, or the vehicle has extraordinary acceleration and braking capabilities to enable it to change to the desired speed at will. Even a Formula 1 race driver racing down a straight stretch of road must brake hard and early to successfully negotiate a sharp bend ahead without skidding off the road, and thereafter accelerate hard down the straight road to reach maximum speed. Therefore, to calculate the realistic maximum model speed that the vehicle is able to travel along any route, not only is it important to determine the road design speed limit to prevent the vehicle from skidding out during a turn, it is also equally important to take into consideration the vehicle's performance capabilities, i.e., the vehicle's maximum speed, as well as its acceleration and deceleration limits.

To determine the maximum model speed for the vehicle at any point along the route of travel, it is necessary to compare the design road speed limit at each point with the maximum vehicle speed limit, and choose the lower of the two. This will set the maximum model speed that the vehicle cannot exceed at every point along the road.

For the purpose of weaponeering assessment, there are generally three classes of vehicles that are of interest. For this thesis, the focus is on the first class of vehicle where the maximum vehicle speed is expected to be around 60 mph. For the vehicle acceleration limit, the time taken for an average vehicle to reach 100 km/h is around ten seconds, and assuming a linear acceleration profile, the acceleration limit is assumed as 0.3g for our study. Assuming that the braking is twice as effective, the deceleration is assumed as 0.6g. The breakdown of the expected vehicle performance for all three classes is summarized in Table 4. With the speed, acceleration, and deceleration limits all established, it is now possible to compute the actual speed of the vehicle at every point along the route of interest.

Table 4. Types of Land Vehicles of Interest

Vehicle type	Max speed ms^{-1} (mph)	Max acceleration (g)	Max braking (g)
Fast	26.82 (60)	0.3	0.6
Slow	13.41 (30)	0.15	0.3
X-slow	6.71 (15)	0.08	0.16

To determine the actual speed of the vehicle at each point along the route, it is important to first identify the points near the current point where the model maximum speed is lower than that for the current point. Imagining ourselves as the driver of the vehicle and assuming we reach the start of the route of interest with a certain reasonable starting speed, at the current instance, we are only concerned with any point lying ahead that requires us to take on a lower speed than our current speed, as these points will determine the time and amount of braking needed to reach the lower speed required in time.

An illustration of the deceleration constraint is provided in Figure 20. The vehicle is traveling at 27 m/s currently, and a point 60 meters ahead has a model maximum speed of 12.4 m/s, and the vehicle is capable of decelerating at 6 m/s^2. Assuming a linear deceleration profile, the relationship between the distance traveled s, the initial velocity u, and the final velocity v is given by

$$v^2 = u^2 + 2as\,. \tag{3.4}$$

Using Equation (3.4), the vehicle is capable of reaching 12.4 m/s from 27 m/s in a distance of 48 meters. Thus, there is no need for the vehicle to reduce speed at the current instance, and it can continue traveling at the model maximum speed until point 2. If the model maximum speed at point 2 is higher, it can even accelerate to 27.4 m/s before decelerating to 12.4 m/s at point 7. Once it reaches point 2, the vehicle can maintain the model maximum speed, and it will need to begin decelerating in order to meet the model maximum speed at point 7 subsequently. For point 3, due to the lower model maximum speed at point 7, the maximum speed possible is only 25.2 m/s, and the vehicle will need to decelerate to this speed when it reaches point 3.

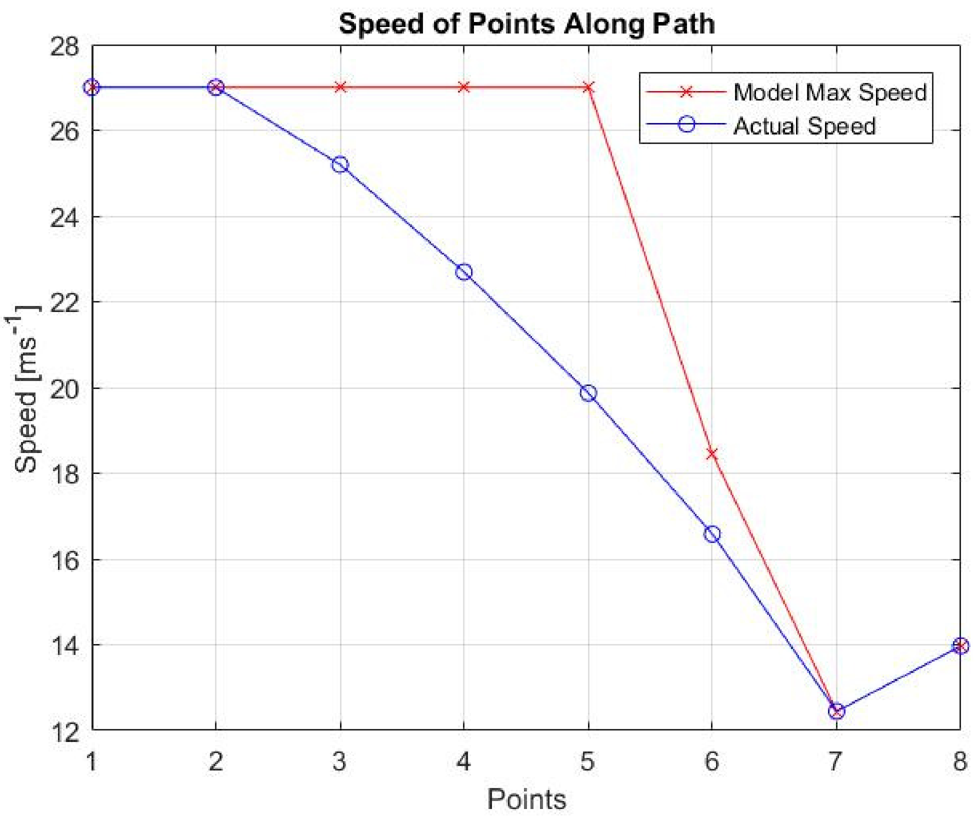


Figure 20. Illustration of Deceleration Constraint in Speed Computation

Thus far, the focus of the example using Figure 20 has been on the effect of the lowest model maximum speed of 12.4 m/s at point 7 on the maximum possible speed at all the points before point 7. Nevertheless, it is important to note that the effects of this for all the points with lower model maximum speeds will need to be considered instead. A model maximum speed of 12.5 m/s at point 6, though still higher than that at point 7, would have a greater impact on the traveling speed at the earlier points due to the shorter stopping distance available. Thus, the model, as it steps through each point on the path, will need to consider all the points with lower model maximum speeds within a specific lookahead distance, and compute the maximum possible speed for the vehicle at the current point based on the deceleration limit and the stopping distance available to each of these possible deceleration "limiting factors."

Along the same line of reasoning, the application for the acceleration limit constraint must also be considered as the vehicle travels from point to point sequentially along the route of interest. For the acceleration constraint, the points of interest are those prior to the current point with lower model maximum speeds. An illustration of the acceleration constraint is provided in Figure 21. which is a continuation of the previous example in Figure 20.

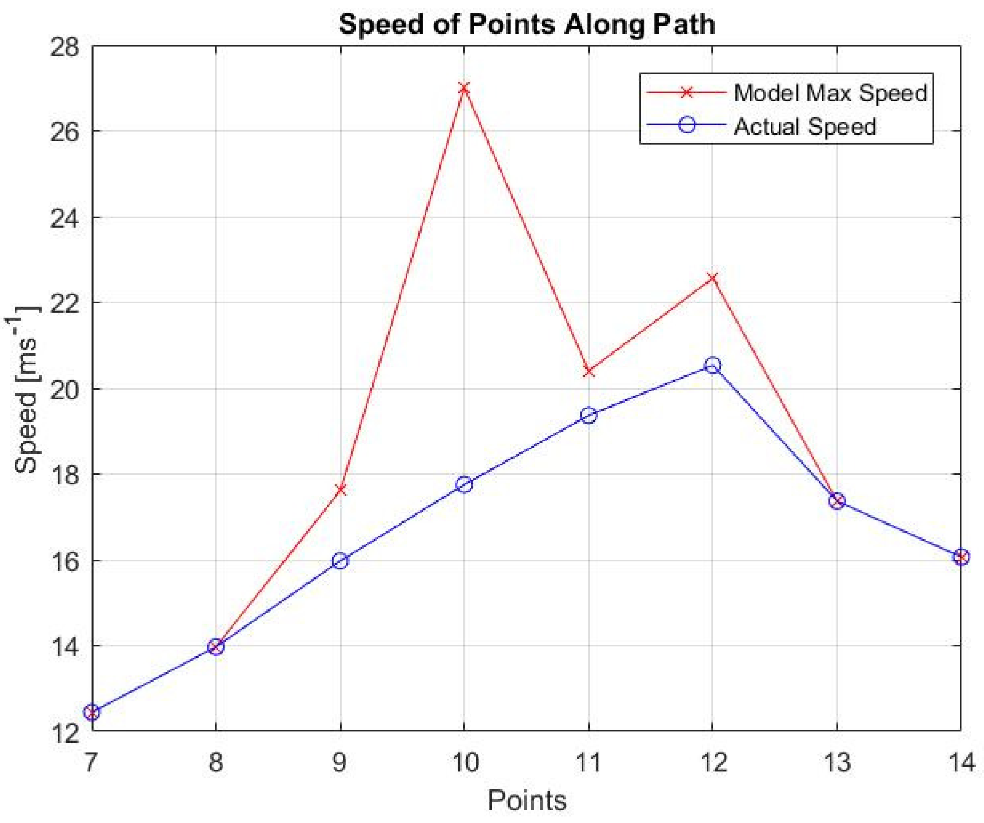


Figure 21. Illustration of Acceleration Constraint in Speed Computation

For point 8, the maximum speed possible due to the acceleration limit is 14.6 m/s; thus, the constraint in this case is due to the model maximum speed. For point 9, the maximum speed of the vehicle at point 7 is only 12.4 m/s, and the vehicle is capable of accelerating at 3 m/s^2. Assuming a linear acceleration profile, using Equation (3.4), the maximum possible speed due to the acceleration limit from point 7 over the 20m distance is 16.5 m/s, which is lower than the model maximum speed of 17.6 m/s. Yet, the realizable speed of the vehicle is lower at 16.0 m/s instead due to the effect of the model maximum speed at point 8, with a pick-up distance of 10 m only.

Thus, the model, as it steps through each point on the path, will need to consider all the points with lower model maximum speeds within a specific lookback distance and compute the maximum possible speed for the vehicle at the current point based on the acceleration limit and the pick-up distance available to each of these possible acceleration "limiting factors."

Having introduced both the deceleration and acceleration constraints for the model, we then need to combine both these constraints for the computation of the actual speed of the vehicle. This is easily achieved by comparing the maximum speeds possible due to both the acceleration and deceleration constraints with the model maximum speed at every point along the path, and choosing the lowest of the three to satisfy all the constraints. All three constraints are presented for the two earlier examples in Figures 22 and 23.

From Figure 22, we can see that the actual speed of the vehicle at points 1–8 is primarily constrained by the deceleration limit and the model maximum speed. This is expected as the model speed limit along this stretch is decreasing in general.

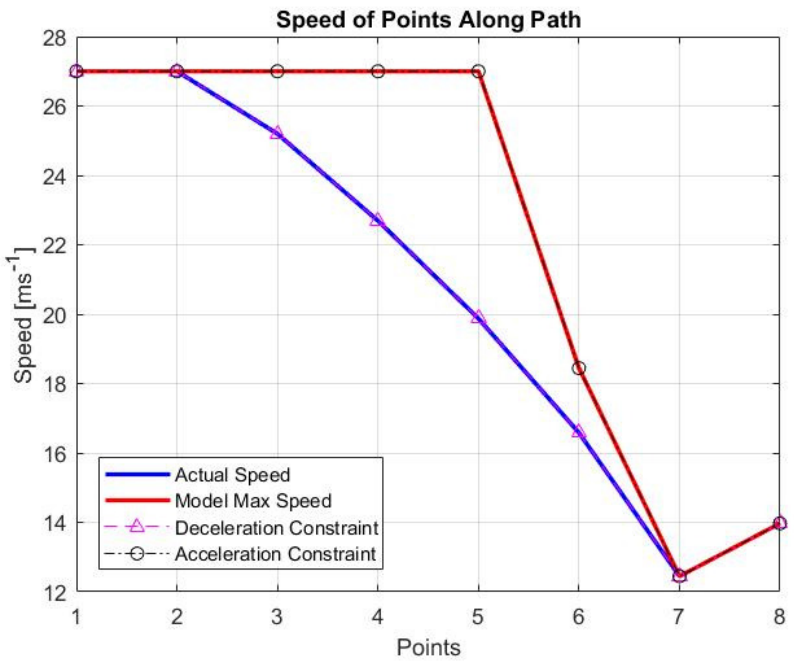


Figure 22. Computation of Speed under All Three Constraints for Points 1–8

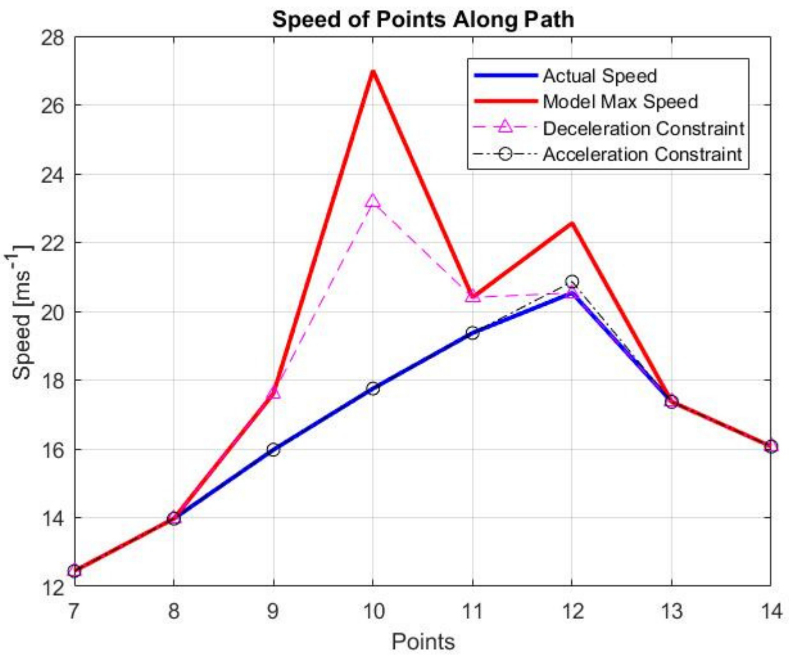


Figure 23. Computation of Speed under All Three Constraints for Points 7–14

From Figure 23, we can clearly see that the speed for the vehicle at points 9 to 11 is determined by the acceleration limits of the vehicle. The computation of the speeds at points 10 and 12 is of interest, as the values of the speed due to the acceleration, deceleration, and model maximum speeds constraints are all distinctly different. To meet the constraints of all three limits, the speed of the vehicle is chosen to be the lowest of the three limits. For point 10, the deciding "limiting factor" is that of the acceleration limit, while that for point 12 is the deceleration limit determined by the low model speed limit at point 13.

The approach that we have discussed thus far is then applied to all the points along the path of interest to generate the highest possible speed of the vehicle at each of the points, thereby generating the velocity profile of the vehicle that satisfies all three physical constraints discussed earlier.

The velocity and acceleration profile for each of the six road types are plotted along the entire path to verify that the speed and acceleration constraints are fully satisfied. Due to the large number of points involved for each plot, a plot of 100 sampling points is provided for each of the road types for better clarity. The respective plots are provided in Figures 24 to 35.

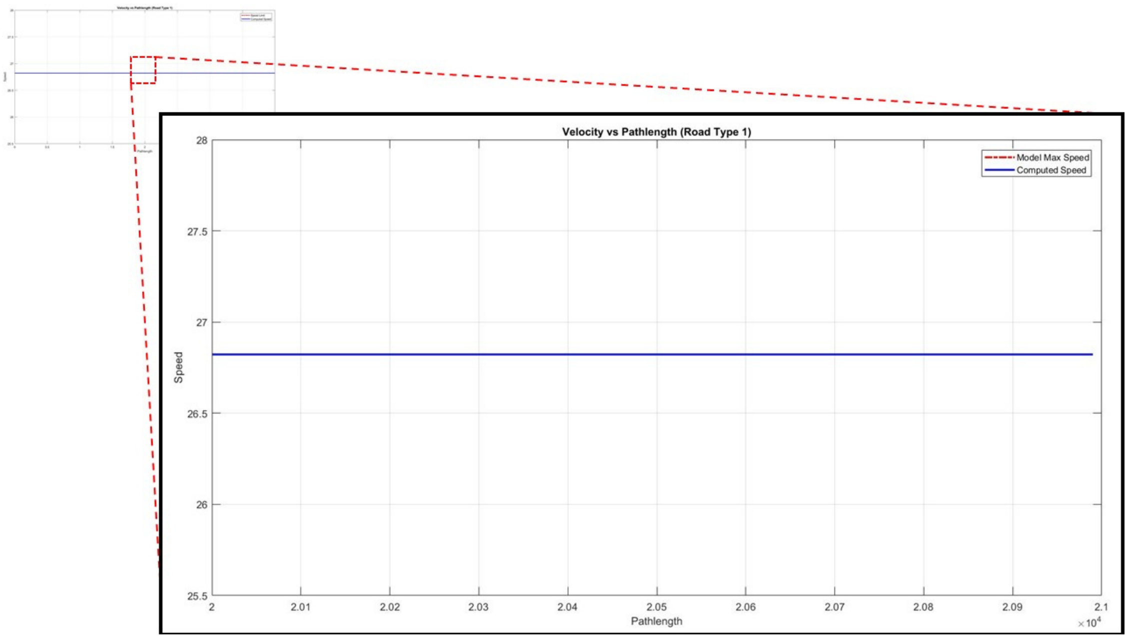


Figure 24. Speed Profile for Road Type 1

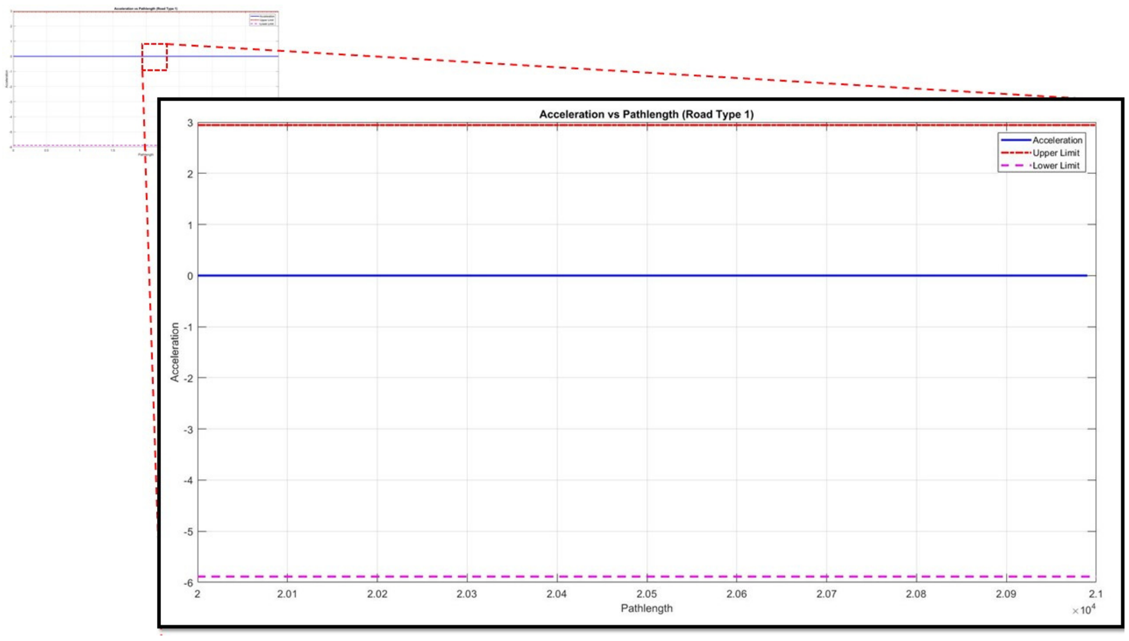


Figure 25. Acceleration Profile for Road Type 1

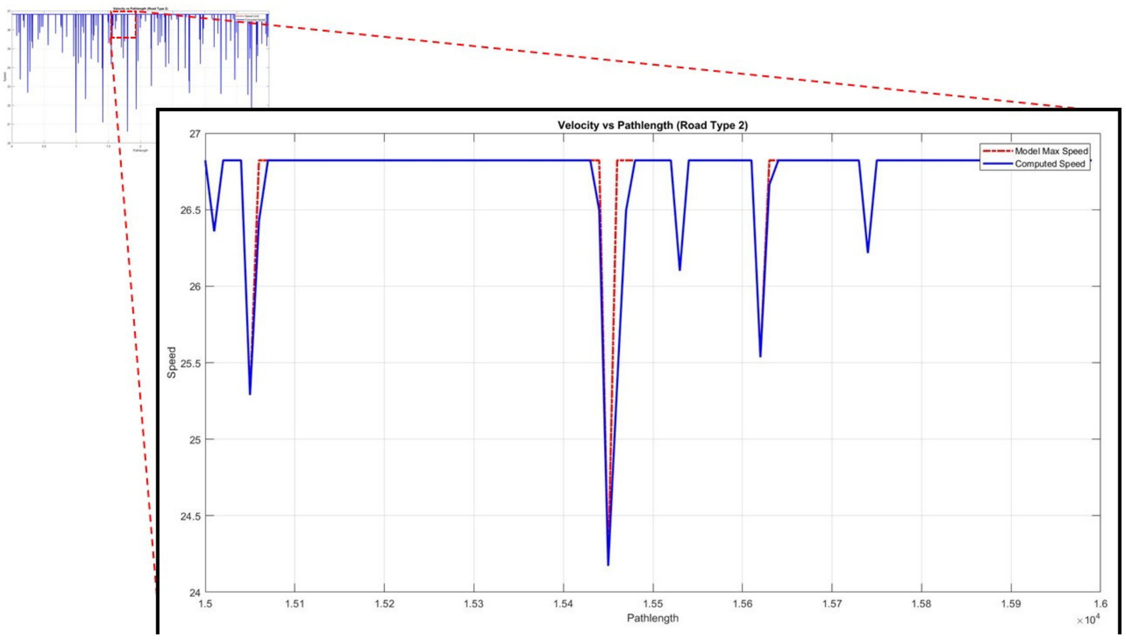


Figure 26. Speed Profile for Road Type 2

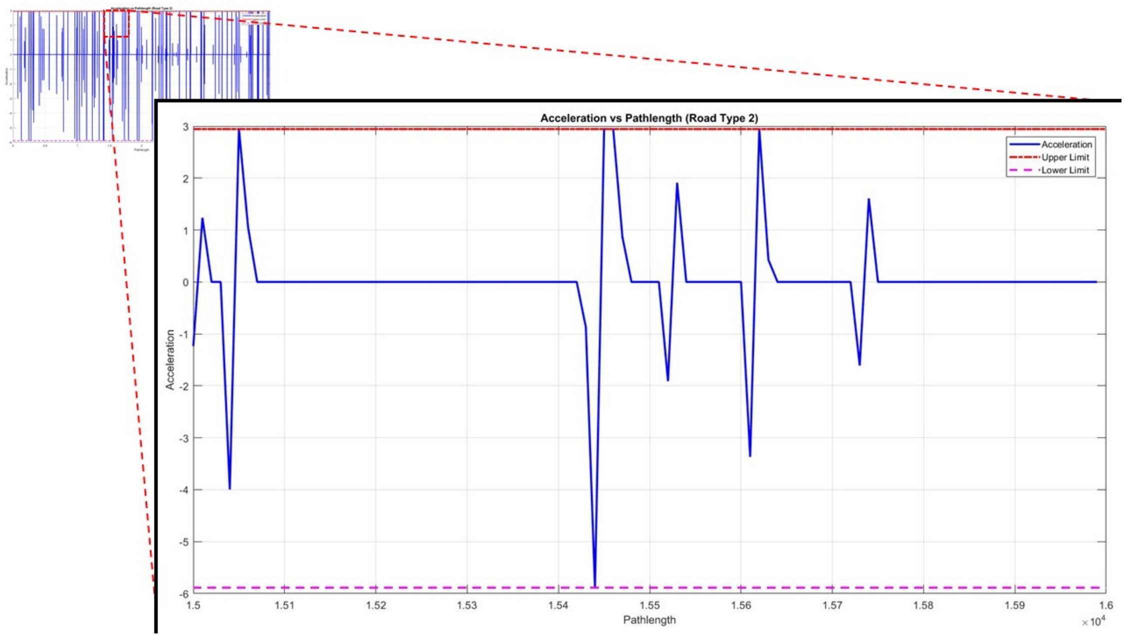


Figure 27. Acceleration Profile for Road Type 2

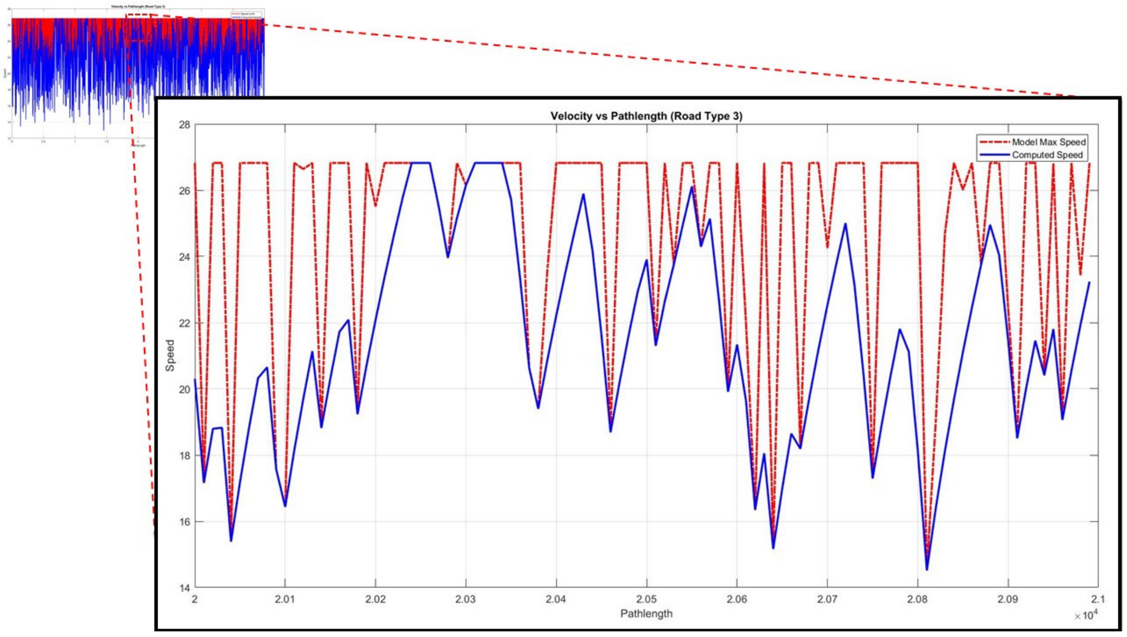


Figure 28. Speed Profile for Road Type 3

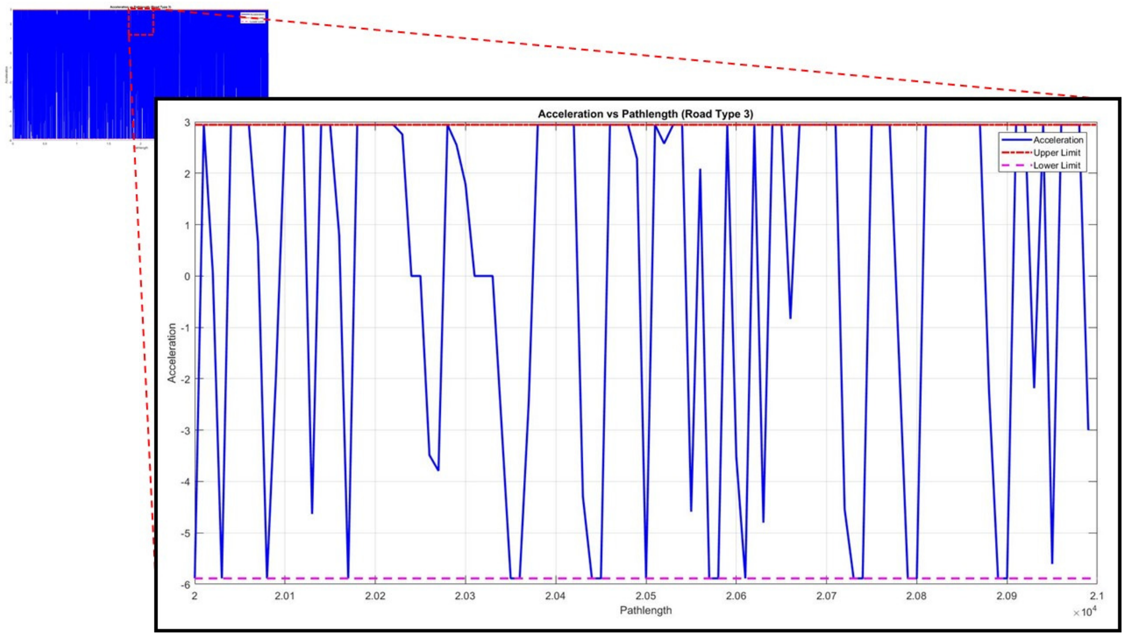


Figure 29. Acceleration Profile for Road Type 3

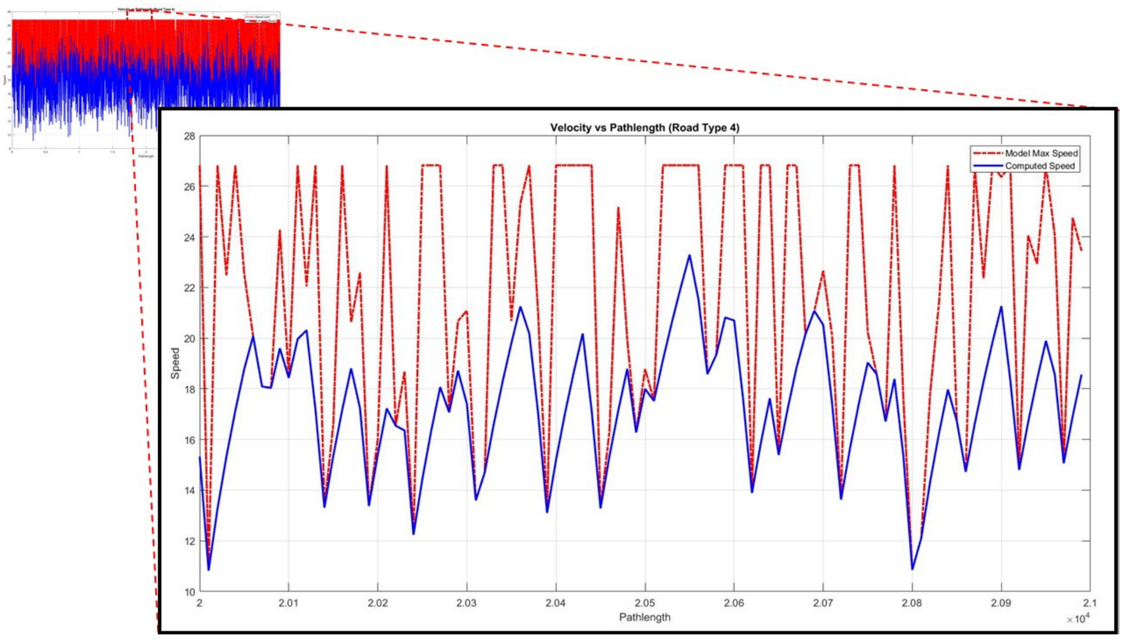


Figure 30. Speed Profile for Road Type 4

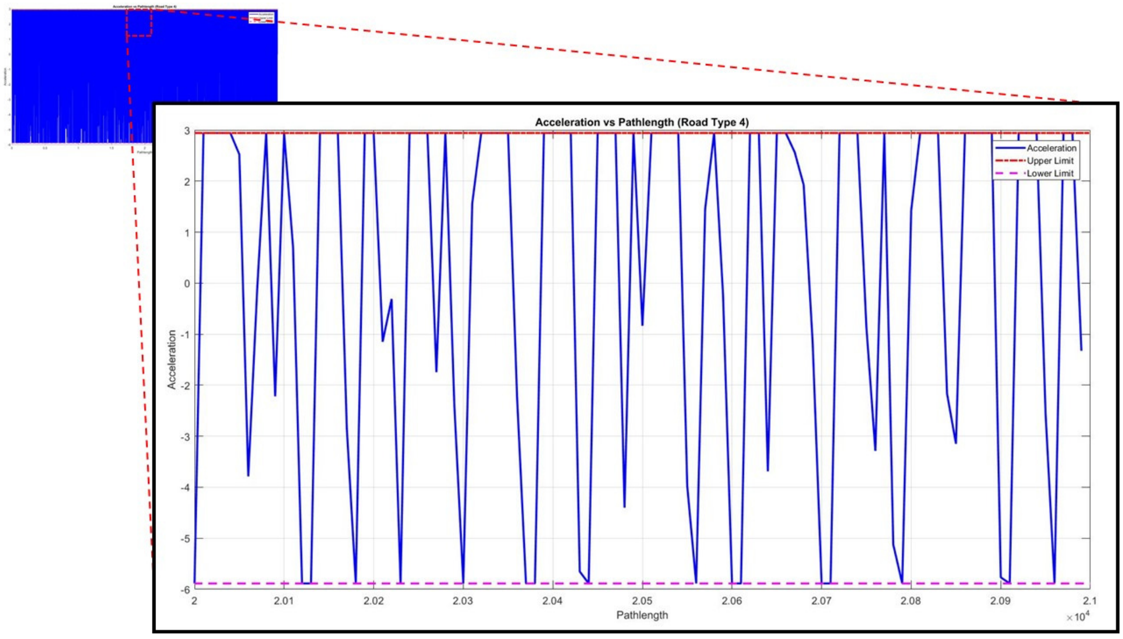


Figure 31. Acceleration Profile for Road Type 4

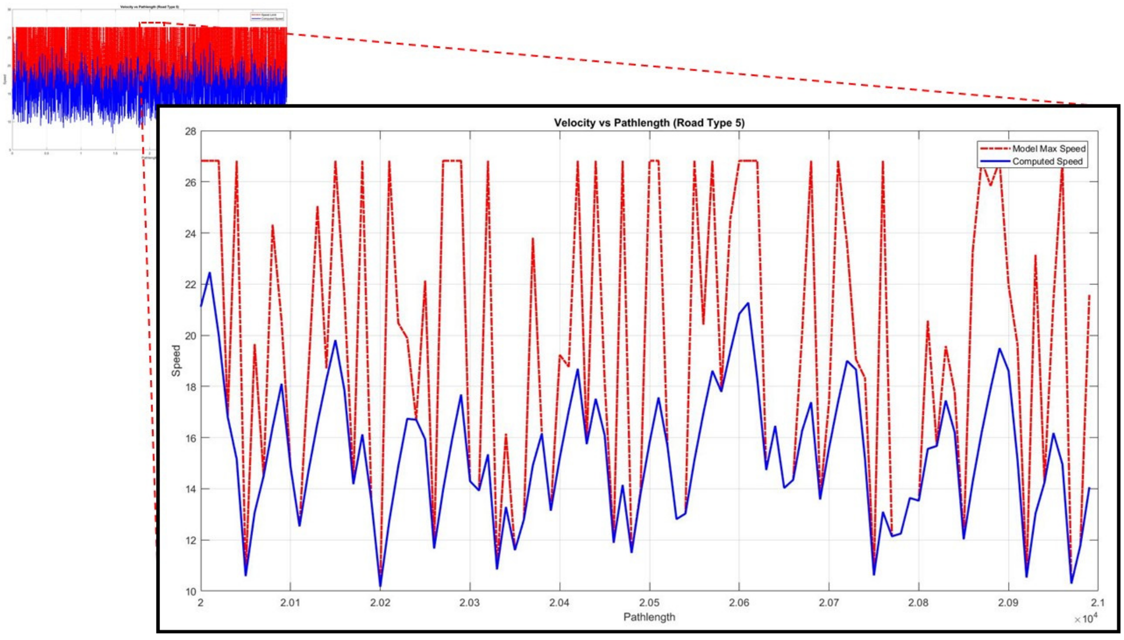


Figure 32. Speed Profile for Road Type 5

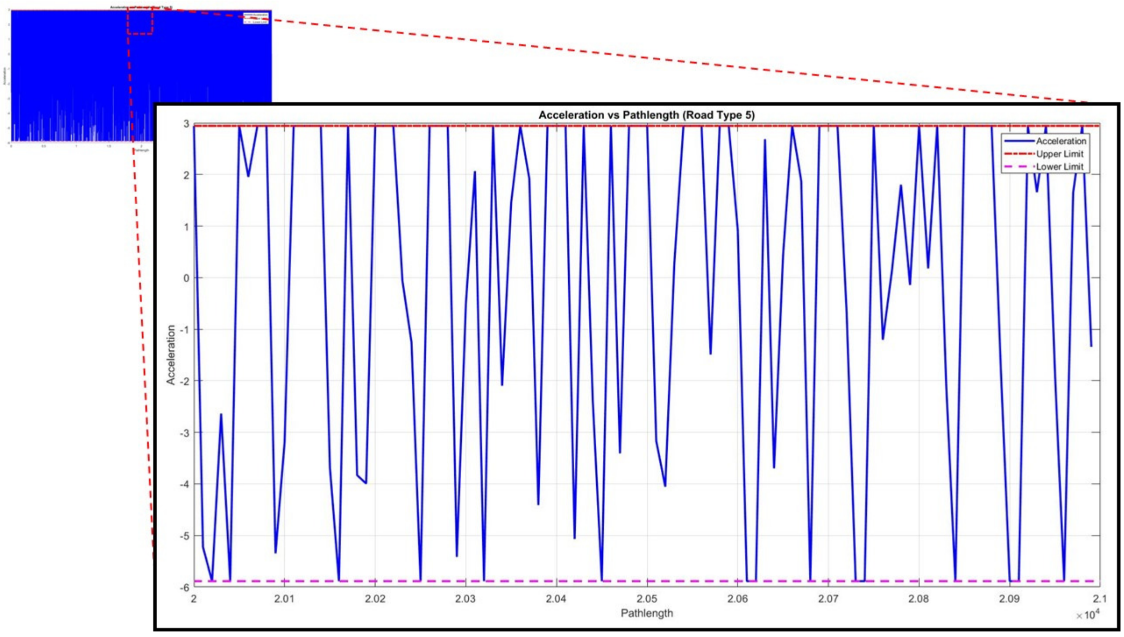


Figure 33. Acceleration Profile for Road Type 5

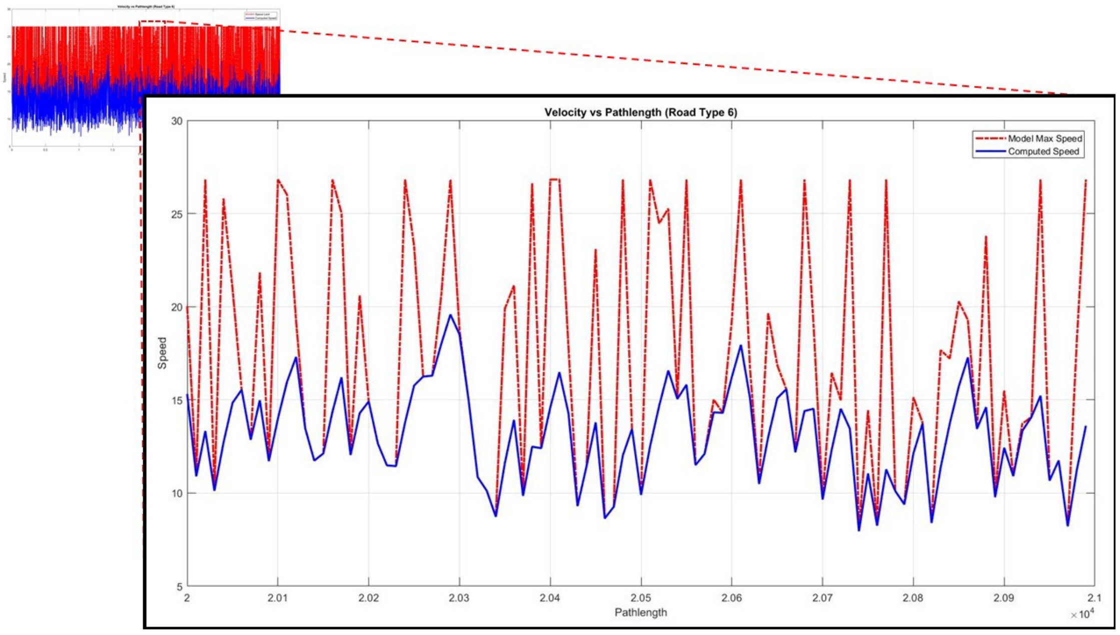


Figure 34. Speed Profile for Road Type 6

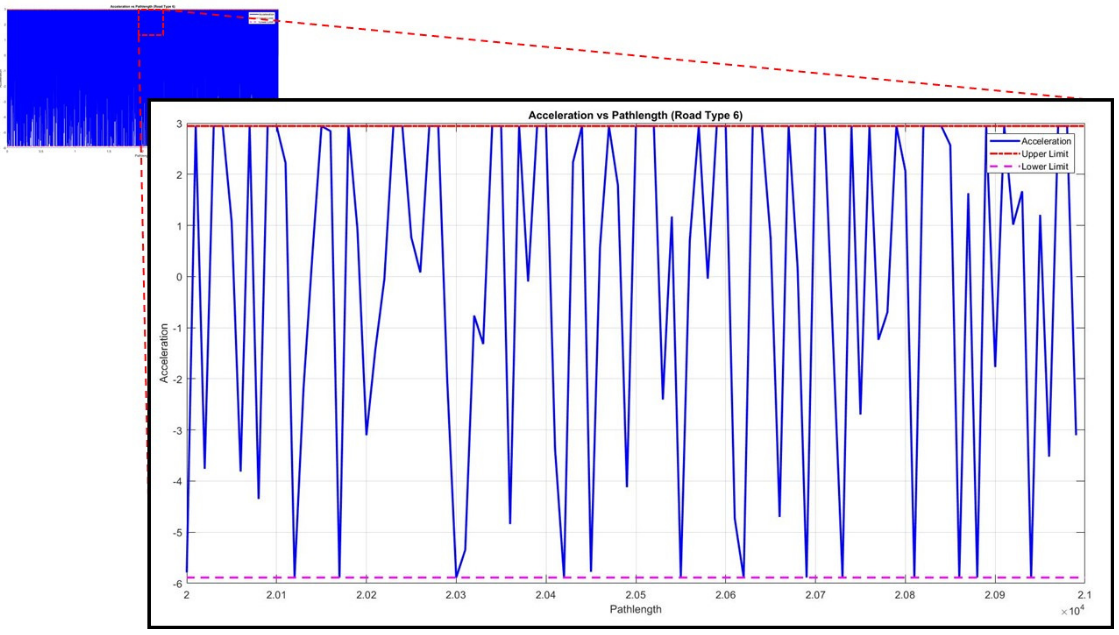


Figure 35. Acceleration Profile for Road Type 6

From all the velocity and acceleration plots presented, we can see that the speeds of the vehicle are kept at less than or equal to the model maximum speeds, while the acceleration of the vehicle does not exceed the limits imposed. This implies that the solution meets all the kinematic constraints satisfactorily; as such, the results obtained from the approach discussed earlier are valid.

During the course of the thesis, an optimization approach is explored to determine the solution for a time-minimization problem. Gill [11] discusses the use of the SNOPT software for the constrained optimization problems, which is applicable to this thesis. Numerous optimization runs are attempted with no meaningful results obtained until the end of the thesis work. As further tuning of the optimization program is required to generate the results that meet all the constraints, the optimization approach is brought to a halt, and further exploration of this area is recommended for future work to obtain the optimal solution to the time-minimization problem.

An alternate approach for the velocity computation is also explored toward the end of the thesis to include the effects of the vehicle dynamics in the computation of the vehicle velocity profile. Kaminer [12] developed a model in the Simulink environment to include the effects of vehicle dynamics on the vehicle trajectory and its velocity profile as it travels down a particular route.

The velocity profile computation is carried out using the model developed by Kaminer [12], and the results for the initial approach and that obtained using the Simulink model are provided in Figure 36. Due to the large number of data points involved, a small sample of the data points is illustrated in the plots for ease of comparison.

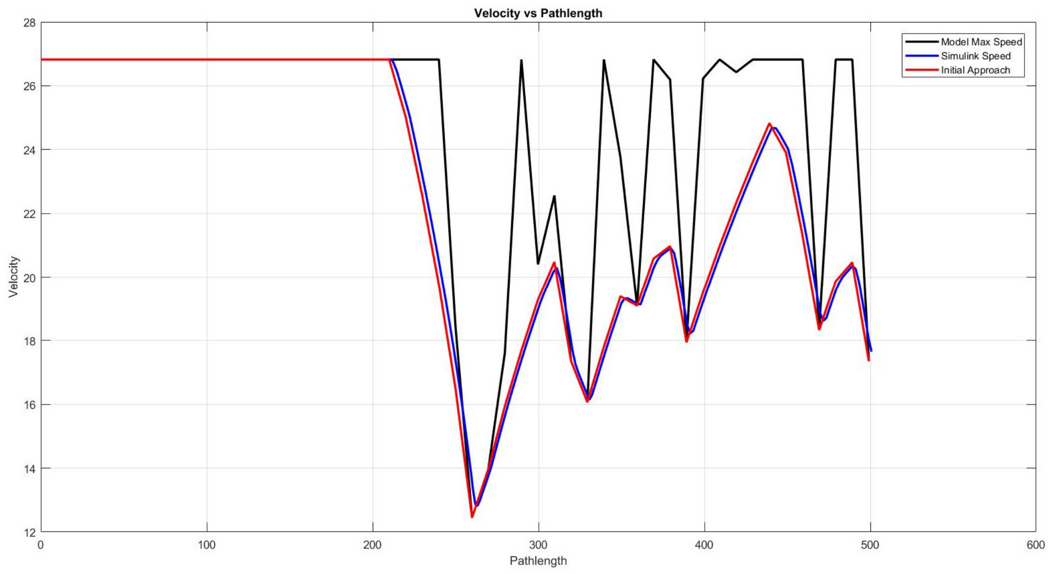


Figure 36. Comparison of Velocity Profile

From the obtained, it can be seen that the velocity profile generated by Simulink does not produce the same sharp changes in velocity that are observed in the velocity profile obtained initially; as such, the Simulink model is able to better capture the effects of vehicle dynamics and produce a more realistic representation of the actual vehicle velocity profile. However, it can also be observed that the velocity profiles generated in both approaches are largely similar. Therefore, the velocity profile generated using the approach discussed earlier provides a very good approximation of the actual vehicle performance under the given constraints.

In view of the limited time remaining for the completion of the thesis work, the Simulink model approach is not studied further. As the results from the initial approach are very close to those obtained from the Simulink model, the original set of velocity profiles computed is retained. For the rest of the thesis, the focus is on using the results from the initial approach used for the computation of the vehicle velocity profile. For future work in similar areas, the Simulink model approach is strongly recommended. A copy of the Simulink model for velocity computation is provided in Appendix B.

B. VELOCITY PROFILE FOR A SEA VESSEL

For the case of the sea vessel, the approach is different, as the simulation model by Aslan [6] is able to generate the velocity profile of the sea vessel throughout the course of its motion. The course of the sea vessel is determined by the rudder inputs, and its maneuverability dependent on the sea vessel characteristics. The sea vessel characteristics that are required as inputs to the simulation model for two examples of sea vessels of interest are summarized in Table 5. Therefore, the aim is to generate a suitable set of rudder input to produce the desired path of travel, shown in Figure 18, for the sea vessel and generate the associated velocity profile.

Table 5. Inputs Characteristics of Sea Vessels. Adapted from [13], [14].

Sea Vessel	Length on Waterline (m)	Displacement (ton)	Maximum Rudder Angle (deg)	Approach Speed (knots)
Bladerunner 51	15	16	30	65
Boston Whaler	7.75	5	30	45

To generate the correct rudder inputs to produce the desired path of travel for the sea vessel, it is necessary to first understand the maneuverability of the sea vessel, which is vessel specific and dependent on the vessel length, weight, and maximum speed. To do that, we first set the rudder angle input at the maximum rudder angle throughout the simulation run. This will simulate the sea vessel making the tightest turn when going at its maximum speed, and from this, we can determine the time taken for the boat to make a complete turn. The results of the maneuverability study are shown in Figure 37.

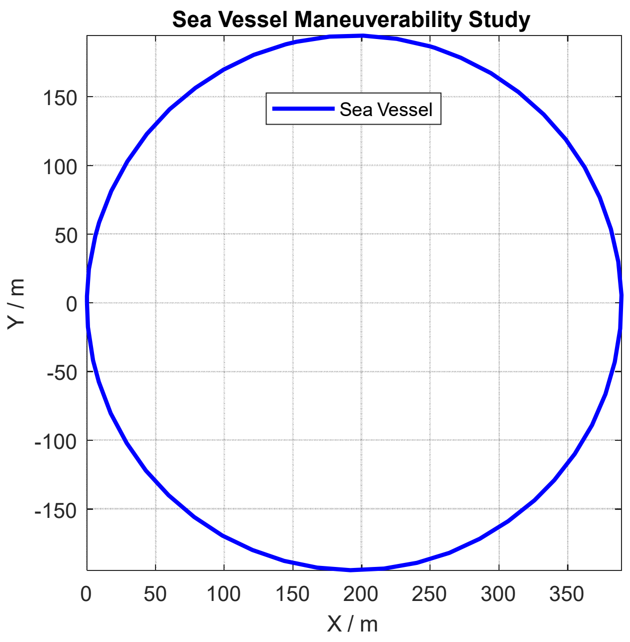


Figure 37. Sea Vessel Maneuverability Study

Through the maneuverability study, we determine that the Bladerunner 51 takes around 41.05 seconds to turn through a complete round of tactical diameter 389 meters at its maximum turning speed of 29.8 m/s. By contrast, the Boston Whaler takes around 13.04 seconds to turn through a complete round of tactical diameter 68 meters at its maximum turning speed of 16.3 m/s. With this information, we can then proceed to work on generating the rudder inputs for producing the path of travel desired.

Since the course of the sea vessel is driven by the rudder inputs, it is necessary to compute the duration of the rudder inputs as the sea vessel goes through each turn. This can be easily done by considering the angle that the ship turns through, since we have already determined the time it takes for the ship to turn through a complete round. Referring to the angle θ in Figure 18, the angle that the sea vessel turns through can be computed by computing $180^o - \theta$, based on the geometrical relationship shown in Figure 38.

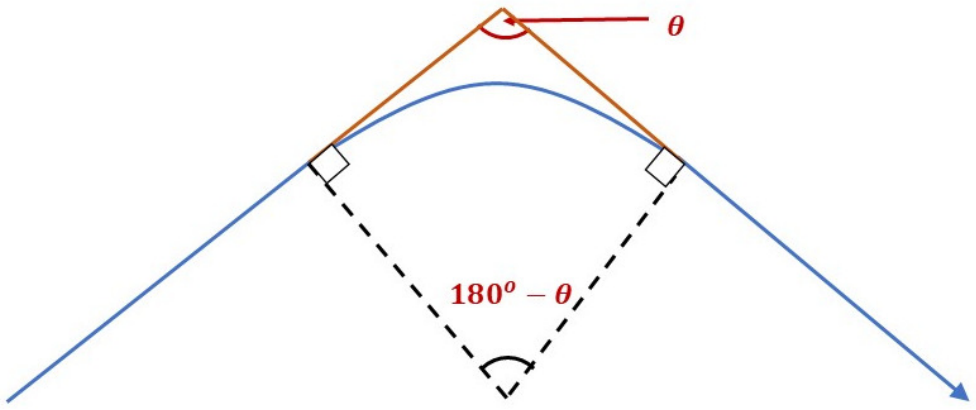


Figure 38. Geometrical Relationship for Sea Vessel Turn Angle

The turn angle computation allows us to compute the turn duration for every turn in the path. Putting all the turn durations into a combined rudder input for the simulation, the control to the system is presented in Figure 39, and the resulting output corresponding to the input is shown in Figure 40.

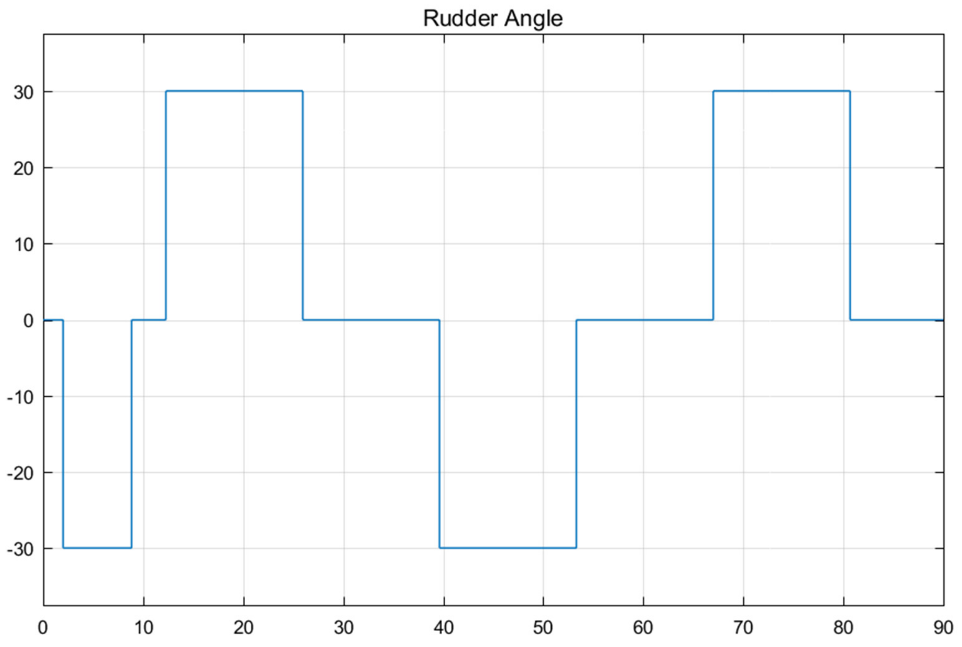


Figure 39. Sea Vessel Rudder Angle Inputs

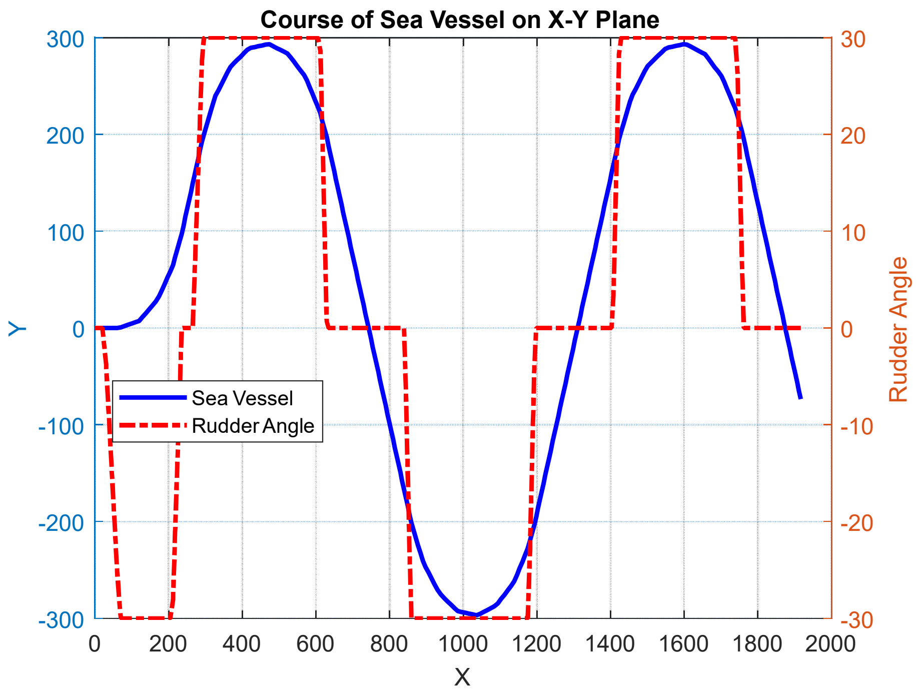


Figure 40. Sea Vessel Simulated Course of Travel

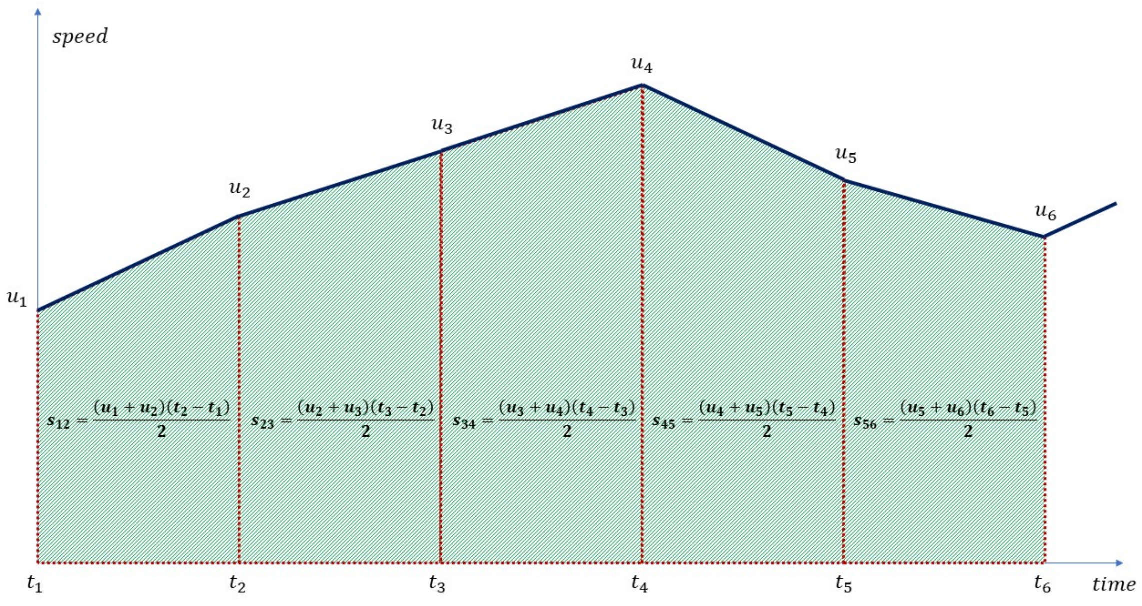
From the results obtained, it can be clearly seen that the sea vessel responds as per intended, turning with the changes in the rudder angles, moving the sea vessel along the sea route that was planned and discussed in the previous chapter.

Having achieved the desired path of travel for the sea vessel, and with the associated velocity profile generated by the simulation model, we are now ready to convert the position and velocity profile into the TSPI format required for the weapon IFS model. A copy of the Simulink model for the sea route simulation can be found in Appendix C.

IV. TIME DISCRETIZATION

A. TRAVELING TIME COMPUTATION

Having computed the position and velocity profiles of both the land and sea vehicle at every point on their paths of travel, we can then compute the time duration for the vehicle to travel between each consecutive pair of points on the path. This can be done using the kinematic model with linear acceleration, as shown in Figure 41.



$speed$
u_1
u_2
u_3
u_4
u_5
u_6
$s_{12} = \frac{(u_1+u_2)(t_2-t_1)}{2}$
$s_{23} = \frac{(u_2+u_3)(t_3-t_2)}{2}$
$s_{34} = \frac{(u_3+u_4)(t_4-t_3)}{2}$
$s_{45} = \frac{(u_4+u_5)(t_5-t_4)}{2}$
$s_{56} = \frac{(u_5+u_6)(t_6-t_5)}{2}$
t_1
t_2
t_3
t_4
t_5
t_6
$time$

Figure 41. Kinematic Model for Time Duration Computation

With the distance between each pair of consecutive points known, the time duration can be computed using the following equation.

$$\triangle t_{i\to i+1} = \frac{2\times s_{i\to i+1}}{u_{i+1}+u_i}. \qquad (4.1)$$

After we have computed the time duration the vehicle spent traveling between points, we can then proceed to recreate the path of travel with points at discrete time intervals to generate the data for the TSPI inputs.

B. CONVERSION TO FIXED-TIME INTERVAL

To recreate the path of travel with points at discrete time interval, the approach is to iteratively step through each point in the path, sum up the time duration for the vehicle to travel to the current point, and check whether the sum exceeds multiples of one second. If the total time duration is less than one second, the model will proceed with the next iterative step to the next point on the path. If the total time duration is more than one second, a new point on the new path with a discrete time interval is created by interpolating between the current point and the previous point for which the time duration is exactly one second, and then subtracting one second from the total time duration. This continues until the total time duration is less than one second, and the model will proceed with the next iterative step to the next point on the path. A visual representation of the iterative process is presented in Figure 42.

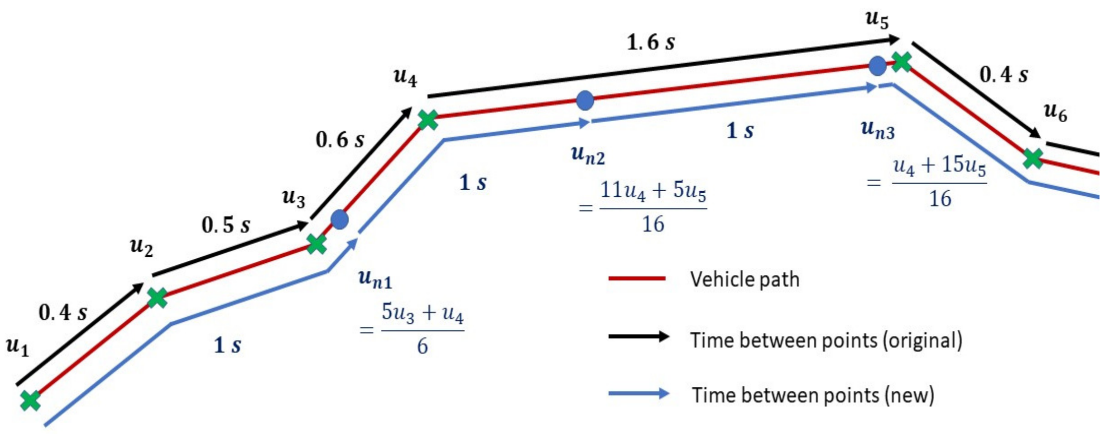


Figure 42. Creation of New Points at Discrete Time Intervals

Besides creating the new points at every discrete time interval, it is important to compute the velocity of the vehicle at each of the newly created points as well. This can be done during the same iterative process by interpolating the velocity of the new point from the velocities of the vehicle at the points between which the new point is being created, based on the position of the new point relative to the positions of the points for the interpolation. This approach is acceptable due to the linear acceleration model that has been adopted for the model earlier. The general equation for the interpolation of the velocity of the new point j from those of point i and $(i+1)$ is given in Equation (4.2).

$$u_j = u_i + \frac{\Delta t_{i\to j}\left(u_{i+1} - u_i\right)}{t_{i+1} - t_i} \tag{4.2}$$

Having derived the position and velocity profile of the new points for the path at each discrete time interval, the required inputs for the TSPI file can now be generated.

C. NEW DATA GENERATION FOR LAND ROUTES

Using the approach that has been discussed, the routes that we have generated previously can now be converted to ones with the discrete time intervals. The new paths for the routes of road type 1 to road type 6 are created and presented in Figures 43 to 48, respectively.

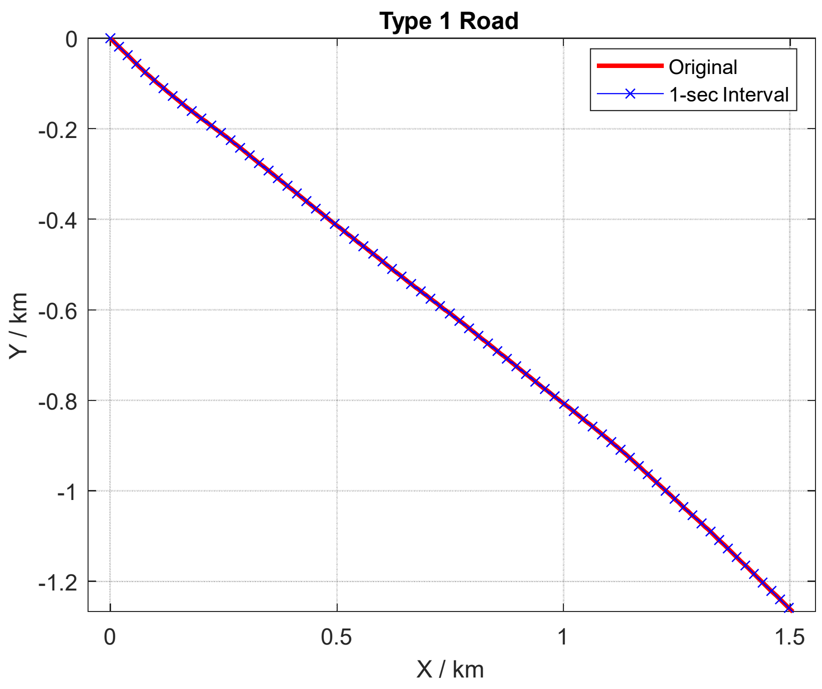


Figure 43. Type 1 Route at Discrete Time Intervals

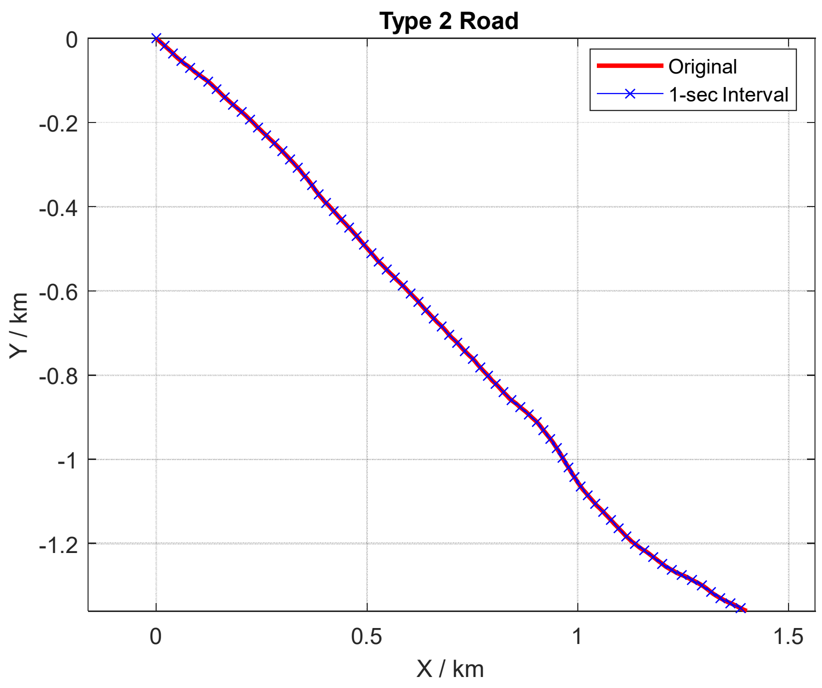


Figure 44. Type 2 Route at Discrete Time Intervals

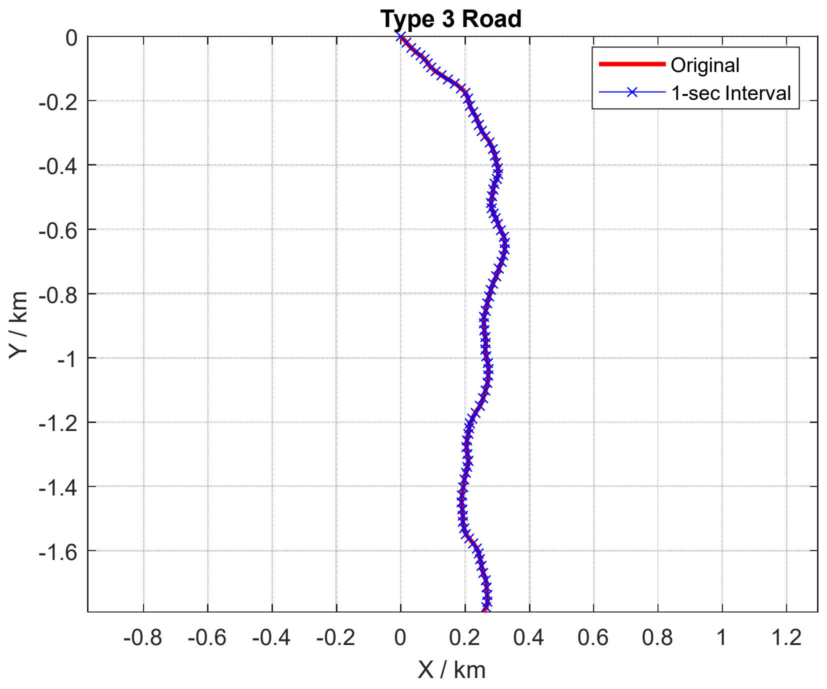


Figure 45. Type 3 Route at Discrete Time Intervals

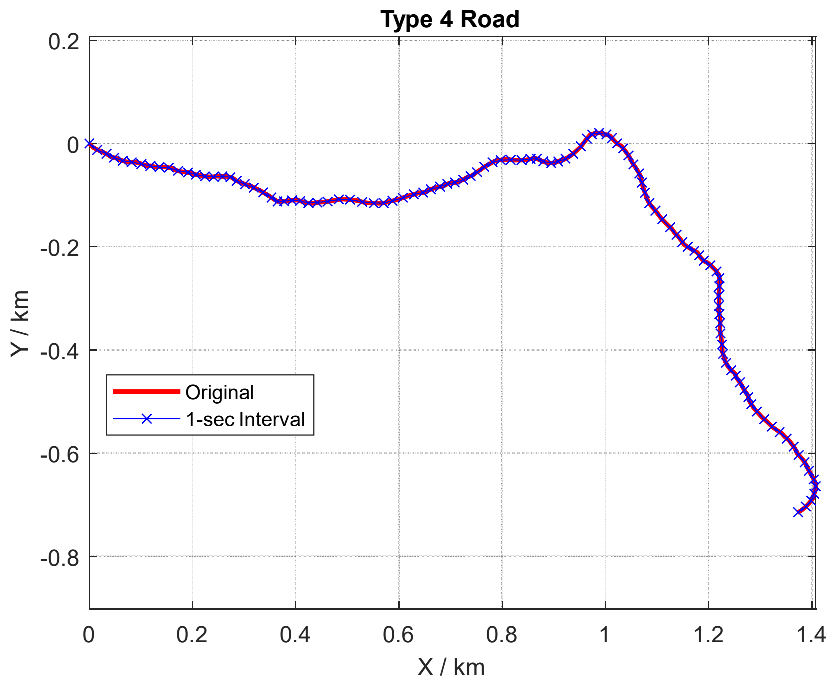


Figure 46. Type 4 Route at Discrete Time Intervals

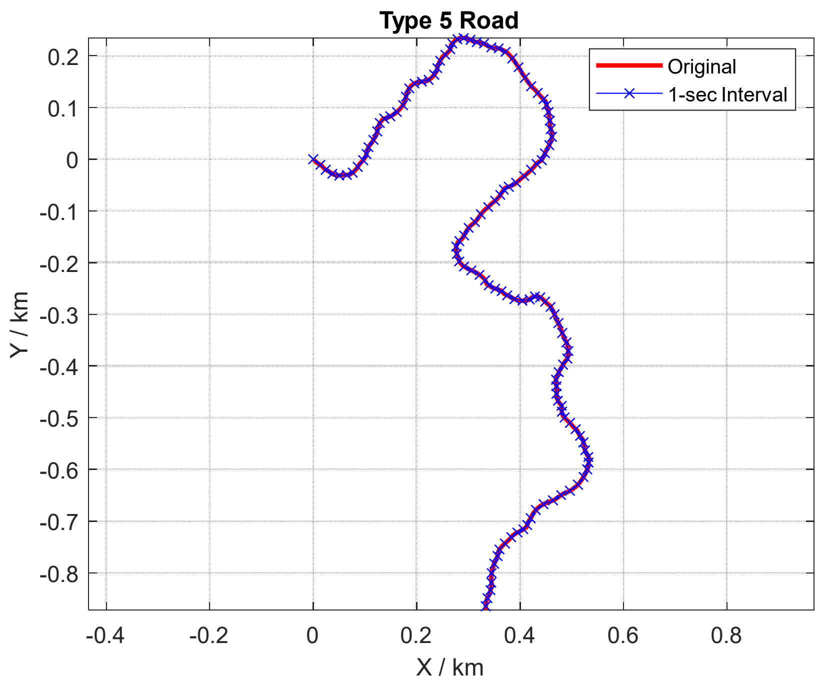


Figure 47. Type 5 Route at Discrete Time Intervals

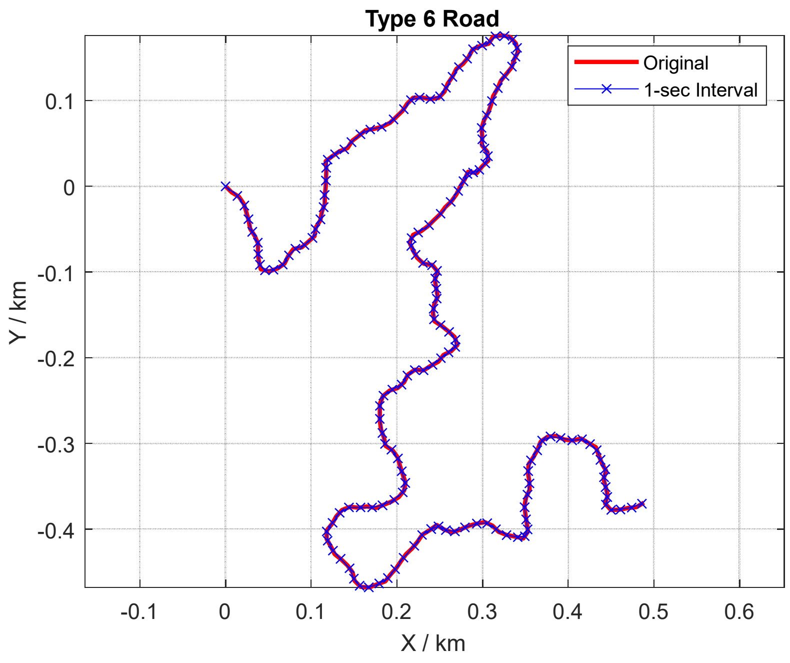


Figure 48. Type 6 Route at Discrete Time Intervals

Due to the large number of data points used for the generation of the six routes for type 1 to type 6, only 5 percent of the points are presented in order to see the new data points on the plots more visibly. Plotting the entire range of data points will only generate plots that look similar to those produced initially.

From the plots obtained, it can be observed that the points get closer together along a curvy portion of the route, and further apart when the route gets straighter. As the vehicle travels faster along a straight road, the amount of distance covered with each second of time is greater. Similarly, when the road gets curvier, the vehicle speed is reduced, and the amount of distance that can be covered in a second of time is reduced accordingly. As such, the results obtained are as expected; therefore, the results are deemed valid and acceptable.

A copy of the MATLAB codes for the computation of the velocity profile and the conversion of the vehicle position along the route to discrete time intervals can be found in Appendix D.

D. NEW DATA GENERATION FOR SEA ROUTES

As the number of points for the path of the sea vessel is significantly lower, the plots of the original path and those of the discrete time interval are presented in Figures 49 and 50, respectively, for ease of comparison.

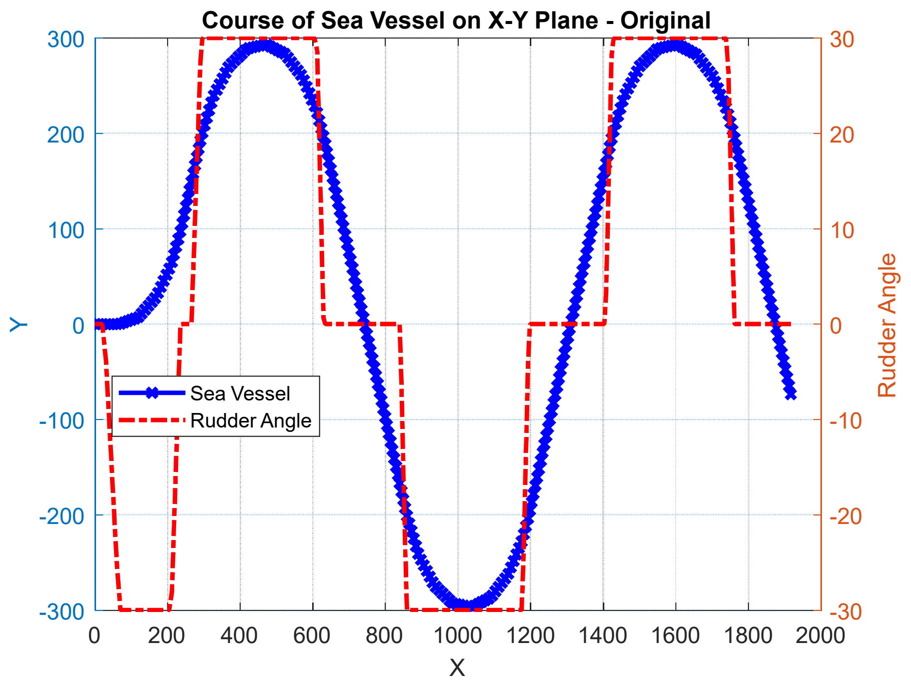


Figure 49. Original Sea Vessel Route

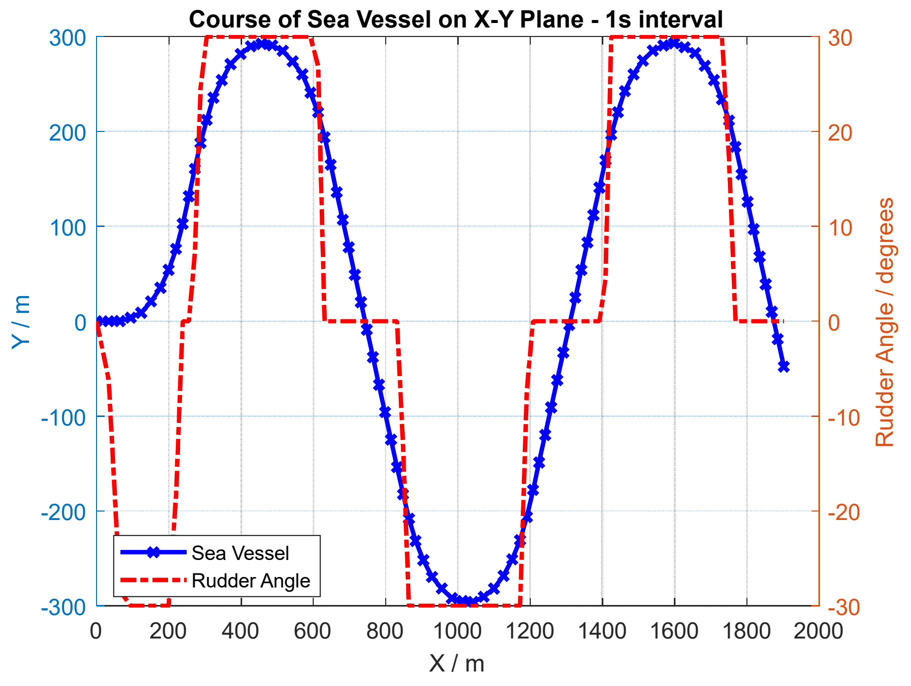


Figure 50. Sea Vessel Route at Discrete Time Interval

Similar to the results obtained for the land vehicle, the data points get closer together as the curvature of the path gets higher, and further apart as the curvature of the path decreases. This is again in accordance with the expectations, and hence the results are valid and acceptable. A copy of the MATLAB codes for the conversion of the sea routes to discrete time intervals can be found in Appendix E.

THIS PAGE INTENTIONALLY LEFT BLANK

V. ROUTE DIGITIZATION

The past few chapters, which constitute the first part of the thesis, focused on generating the routes of interest for weapon assessment study and converting them into the TSPI inputs that can be used in the weapon IFS model for analysis. For the rest of the chapters, which constitute the second part of the thesis, the aim is to come up with a methodology to convert an actual route selected from a map and convert it into a digital form for route characterization and weapon assessment. An additional requirement for this methodology is a Visual Basic for Applications (VBA) environment. This environment enables intended users to utilize the program in a restricted work area with no access to advanced software for performing such analysis.

A. SELECTING THE ROUTE OF INTEREST

For the route selection, we want to take an approach that is simple, user-friendly, and easily accessible. With that in mind, we start off with the selection of the map route using Google Maps [15], a web-based electronic map, commonly used by many around the world. Google Maps is easily accessed using web browsers and an active internet connection. The limitation to this approach is that the routes available for selection are limited to areas that Google has managed to access and map out.

Selection of a route using Google Maps [15] is simple and straightforward, as this can be done using the "Directions" function that can be accessed with just a click of a mouse button. Thereafter, the selection of the route of interest can be done by clicking on the locations on the map to specify the start and end points of the route. Google Maps [15] will then process and generate the possible routes to get from the start point to the end point. Then, users can choose from the routes that Google Maps [15] has suggested or can specify the route that they want by including additional waypoints. An illustration of this for a route along the Carmel Valley road is presented in Figure 51.

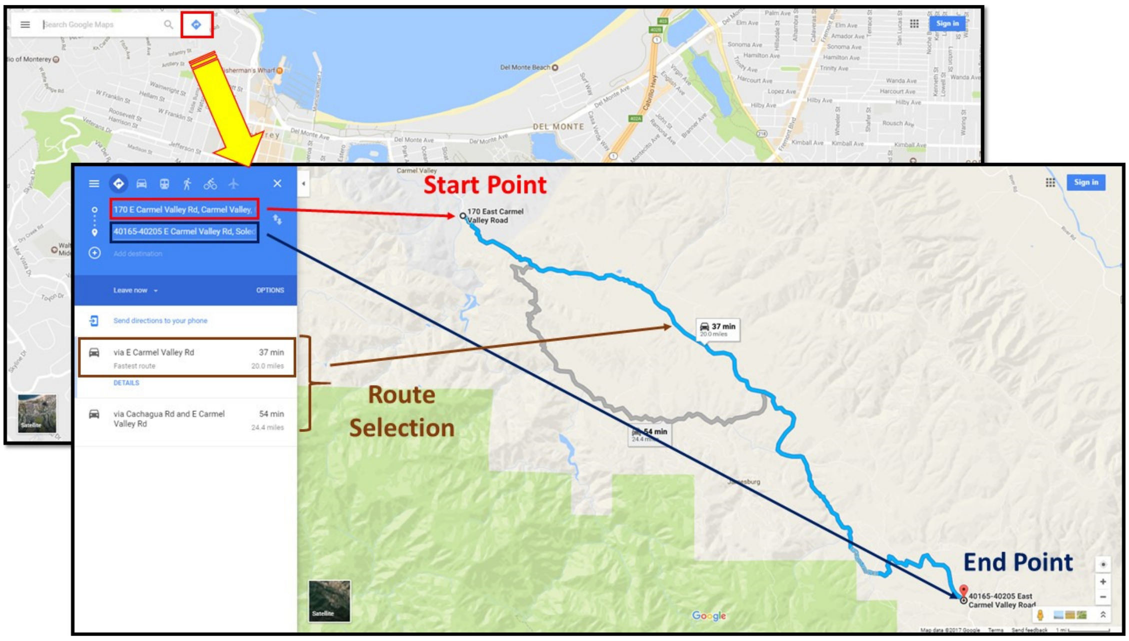

Start Point
170 East Carmel Valley Road
Route Selection
End Point
40165-40205 East Carmel Valley Road
170 E Carmel Valley Rd, Carmel Valley,
40165-40205 E Carmel Valley Rd, Soled
via E Carmel Valley Rd
37 min
Fastest route
20.0 miles
via Cachagua Rd and E Carmel Valley Rd
54 min
24.4 miles

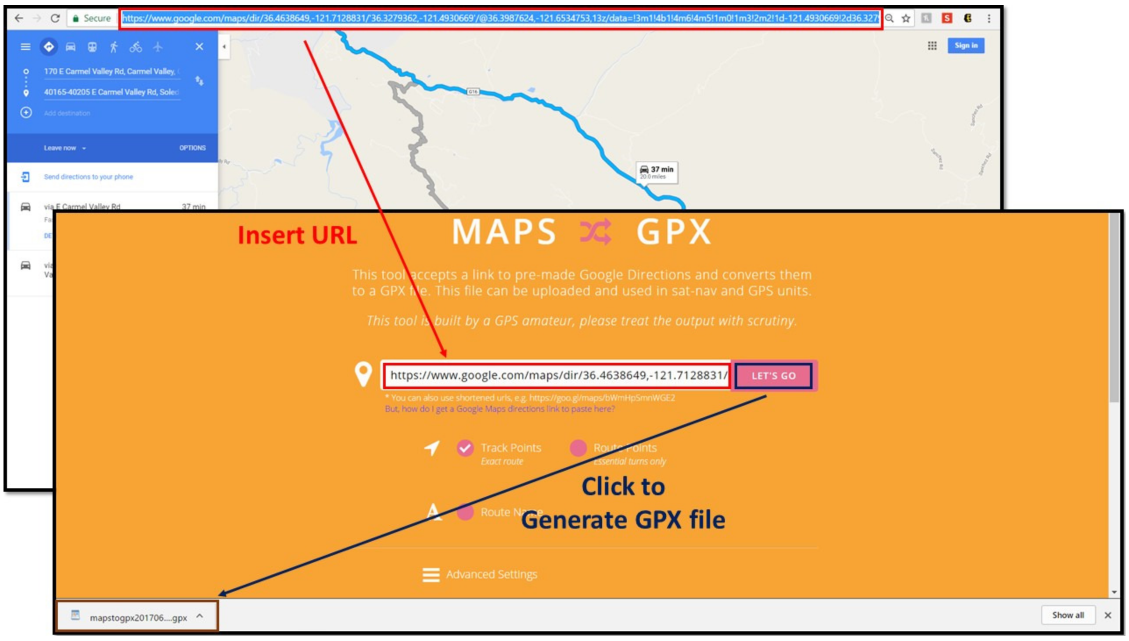
Figure 51. Selection of Route of Interest

After selecting the route of interest, we can then proceed to extract the route information and convert it into a digital form that allows for further processing of the information that we are keen to sieve out.

B. IMPORTING THE ROUTE OF INTEREST

One of the common file formats for presenting map information is the GPX file format that GPS devices utilize. For the purpose of this thesis, we use the GPX file format for the extraction of the route information. While Google Maps [15] provides a user-friendly interface for route selection, it does not allow for the extraction of the route information. As such, we need to utilize other avenues to get the data we need from Google Maps. A quick search on the web will reveal many tools, available for purchase or at no cost, for performing this task. With the key consideration of accessibility in mind, a web-based conversion tool is preferred. Therefore, for this study, a third-party website, MapsToGPX [16], is chosen not only for its accessibility online but its simplicity of use.

Extraction of the map information can be easily done by inserting the Uniform Resource Locator (URL), also commonly known as the web address, of the route that we have selected in Google Maps [15] earlier and clicking the "Let's Go" button to process the route information into a downloadable GPX file. An illustration of the usage of the map conversion tool is provided in Figure 52. With this file, the information of all the points that define the route selected in Google Maps is retrieved in digital format.


https://www.google.com/maps/dir/36.4638649,-121.7128831/'36.3279362,-121.4930669'/@36.3987624,-121.6534753,13z/data=!3m1!4b1!4m6!4m5!1m0!1m3!2m2!1d-121.4930669!2d36.327

170 E Carmel Valley Rd, Carmel Valley,

40165-40205 E Carmel Valley Rd, Soled

This tool accepts a link to pre-made Google Directions and converts them to a GPX file. This file can be uploaded and used in sat-nav and GPS units.

This tool is built by a GPS amateur, please treat the output with scrutiny.

https://www.google.com/maps/dir/36.4638649,-121.7128831/

* You can also use shortened urls, e.g. https://goo.gl/maps/bWmHpSmnWGE2

But, how do I get a Google Maps directions link to paste here?

Figure 52. Generation of Route Data in GPX Format

After obtaining the route information in GPX file format, we must import the information into a software tool so that the information can be read and processed easily. For this purpose, both MATLAB and Microsoft EXCEL are capable of importing the GPX file and presenting the map data in terms of their latitude and longitude coordinates in array form. With this, we are ready to process the information for further analysis of the route.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. ROUTE PROCESSING AND ANALYSIS

Using the approach discussed earlier under “Map Coordinates Transformation” in Chapter I, we can convert the map data extracted from the GPX file from the geodetic latitude and longitude coordinates to the ENU coordinate system. In this way, the Easting and Northing coordinates are equivalent to the familiar Cartesian coordinate system.

The key parameter that we want to extract from the route is the standard deviation of the change in heading of the route. This parameter enables us to classify the route as one of the six road types presented in Table 2 in Chapter II. To compute the standard deviation of the change in heading, first we need to standardize the route make-up for a fair statistical computation of the heading and change in heading. This is done by converting the route from one comprising irregular route points to one with regularly spaced route points, and for this thesis, the route points are spaced ten meters apart.

A. CONVERSION TO REGULAR INTERVALS

For the conversion of the route points to regular intervals, the approach is similar to that used for conversion to the fixed-time interval that we discussed in Chapter IV.

To recreate the path of travel with points at equal intervals of ten meters apart, the process required is to iteratively step through each point in the path, sum up the distance traveled for the vehicle to reach the current point, and check whether the sum exceeds multiples of ten meters. If the total distance traveled is less than ten meters, the model will proceed with the next iterative step to the next point on the path. If the total time duration is more than ten meters, a new point on the new path with regularly spaced points is created by interpolating for the position between the current point and the previous point for which the distance traveled is exactly ten meters, and then subtracting ten meters from the total distance traveled. This continues until the total distance travelled is less than ten meters, and the model will proceed with the next iterative step to the next point on the path. A visual representation of the iterative process is presented in Figure 53.

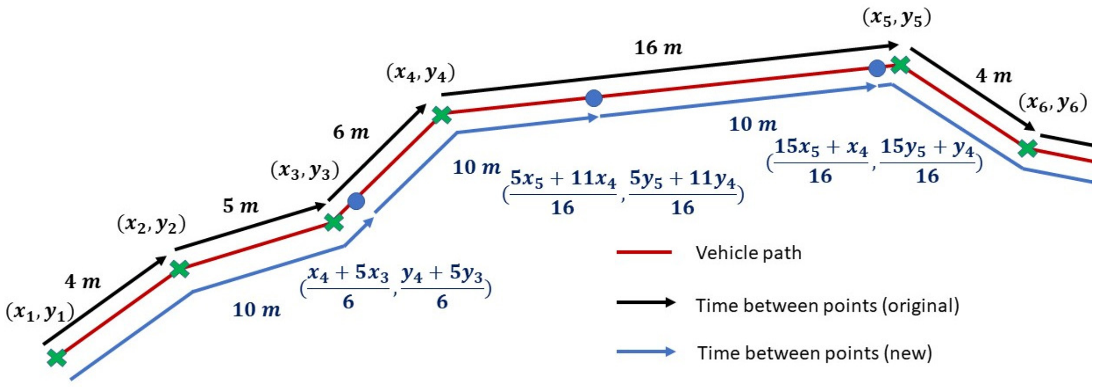

(x_1, y_1)
4 m
(x_2, y_2)
5 m
(x_3, y_3)
6 m
(x_4, y_4)
16 m
(x_5, y_5)
4 m
(x_6, y_6)
10 m
$(\frac{x_4+5x_3}{6}, \frac{y_4+5y_3}{6})$
10 m
$(\frac{5x_5+11x_4}{16}, \frac{5y_5+11y_4}{16})$
10 m
$(\frac{15x_5+x_4}{16}, \frac{15y_5+y_4}{16})$
Vehicle path
Time between points (original)
Time between points (new)

Figure 53. Creation of New Points at Regular Distance Interval

The general equation for the interpolation of the x and y coordinates of the new point j from those of point i and $(i + 1)$ is given as

$$x_j = x_i + \frac{\Delta s_{i\to j}\left(x_{i+1} - x_i\right)}{s_{i+1} - s_i} \tag{6.1}$$

$$y_j = y_i + \frac{\Delta s_{i\to j}\left(y_{i+1} - y_i\right)}{s_{i+1} - s_i} \tag{6.2}$$

This approach is used to convert the Carmel Valley route extracted from Google Maps [15] into a route made up of points spaced ten meters apart. The original and newly converted routes are presented in Figures 53 and 54, respectively.

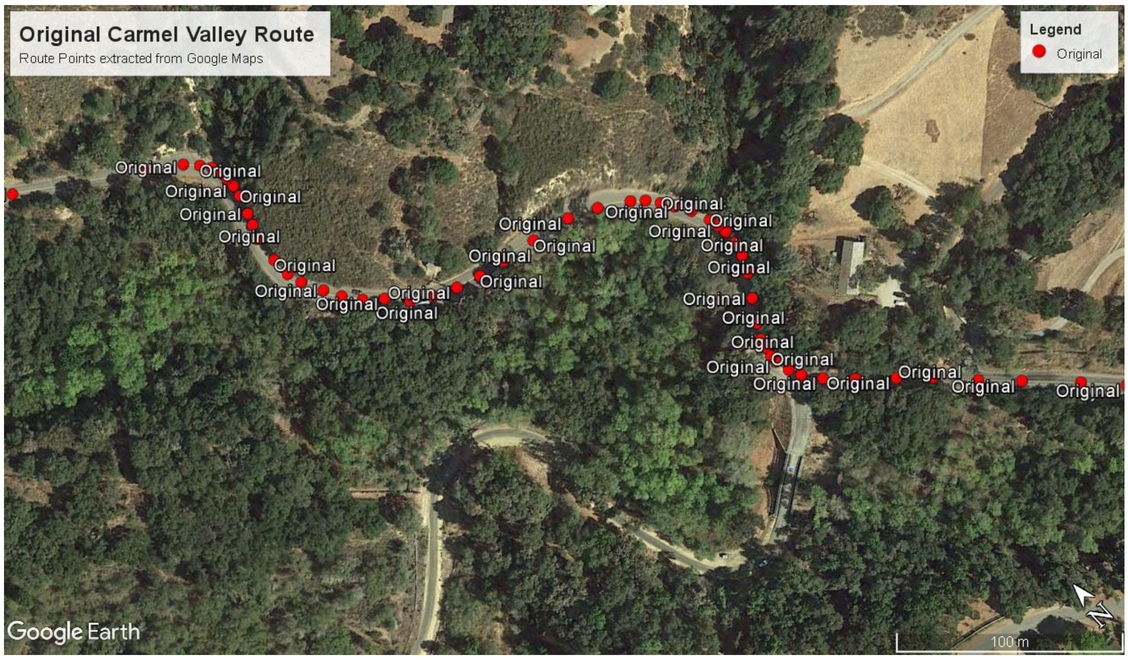

Original Carmel Valley Route
Route Points extracted from Google Maps
Legend
● Original
Original
Google Earth
100 m

Figure 54. Original Route Points Imported

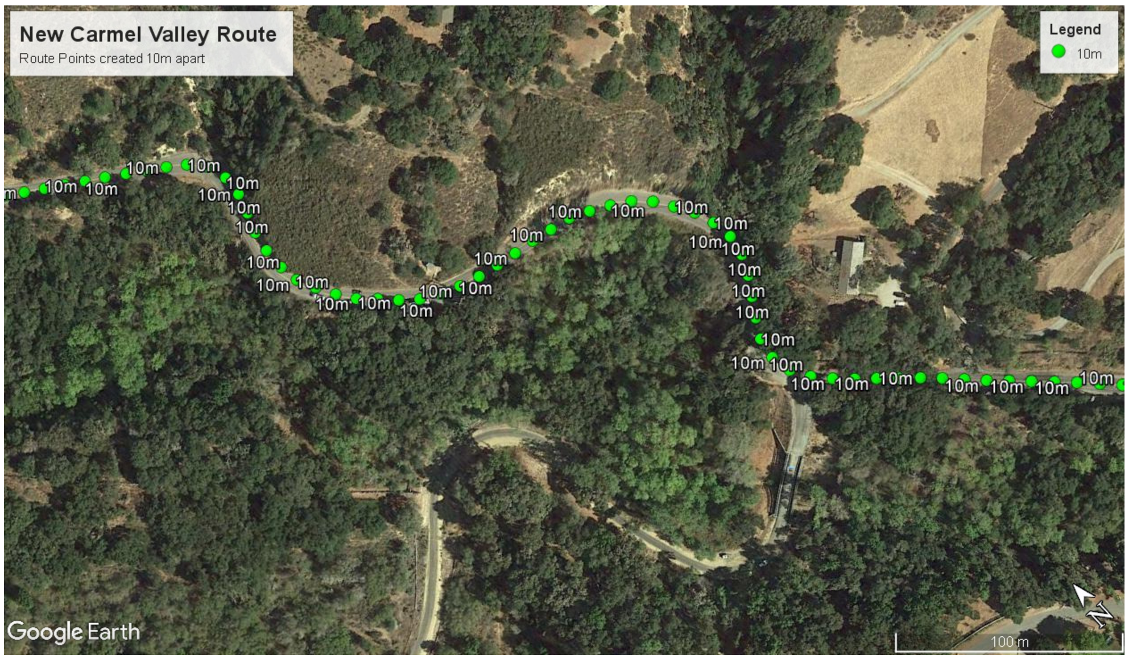


Figure 55. New Regularly Spaced Route Points

From the comparison of the visual plots of the two routes, we can see that the original route has been successfully converted into one with the route points regularly spaced at ten meters apart. Having established the methodology for standardizing the route in terms of their make-up, we can then proceed to conduct the statistical analysis for the converted route.

B. CHANGE OF HEADING COMPUTATION

To compute the heading, we need to establish the heading of each of the points on the path. We do this by calculating the slope of the line joining the points immediately before and after the point of interest. After the standardization of the route points earlier, the regularly-spaced route points allow this approach to give a good approximation for the slope of the tangent to the route at the current point. This methodology is applied for the computation of the slope at every point along the route. The computation of the slope of the tangent to the route and the heading at a route point is presented in Figure 56.

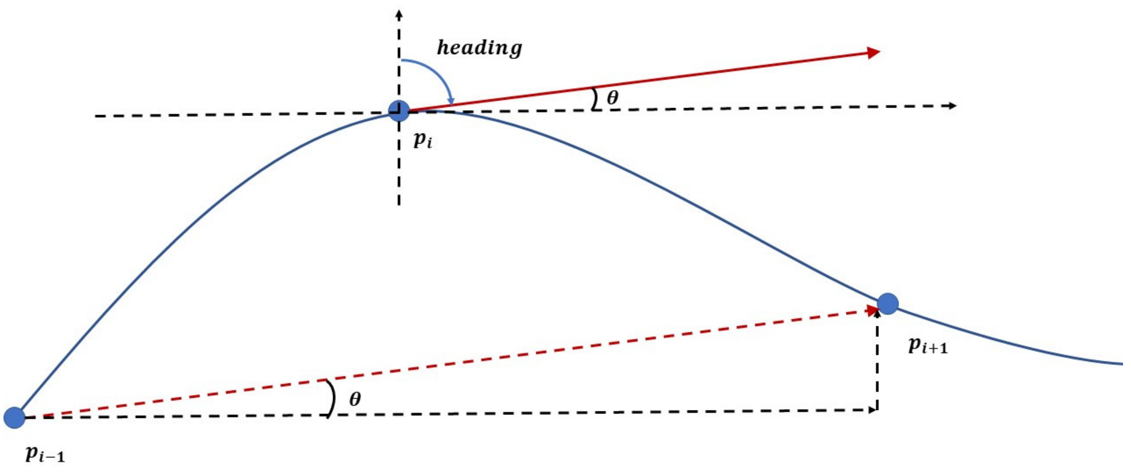


Figure 56. Computation of Heading

With this approach, the heading of every point, less the start and end points, along the route can be computed. With the heading on each point established, the change of heading can then be easily computed by taking the difference between the heading of the current point and that of the point prior. The computation of the change in heading of the point is illustrated in Figure 57.

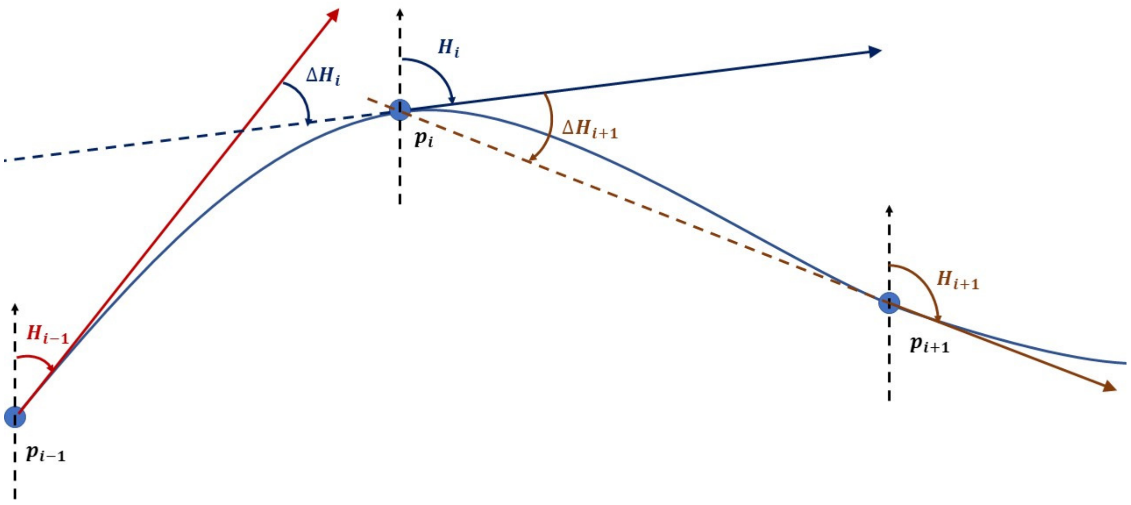


Figure 57. Computation of Change in Heading

The approach is tested on a short winding route, shown in Figure 58. The results of the heading computations are presented in Figures 59 and 60.

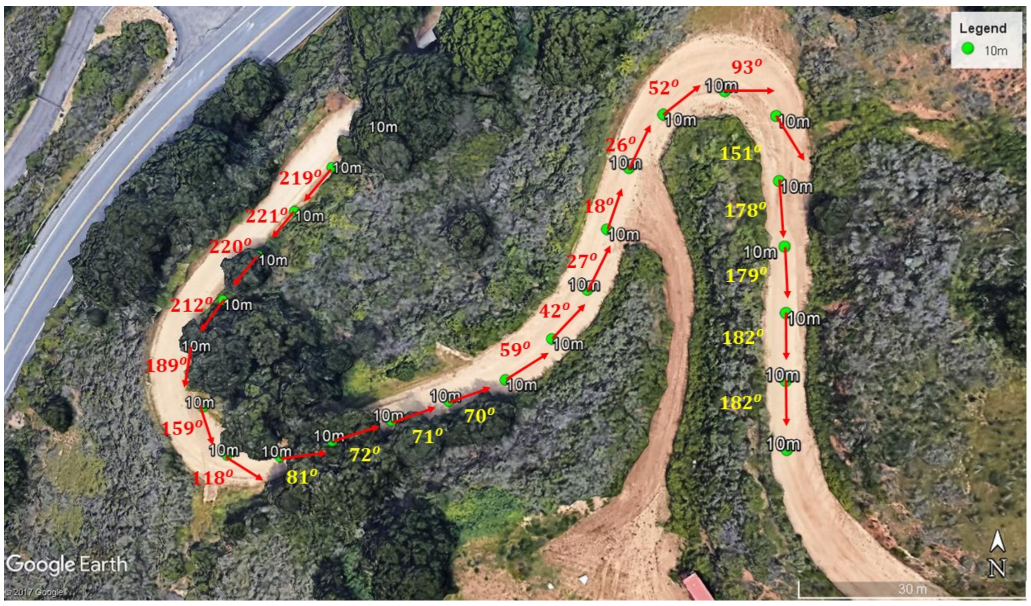


Figure 58. Regularly Spaced Points on Test Route

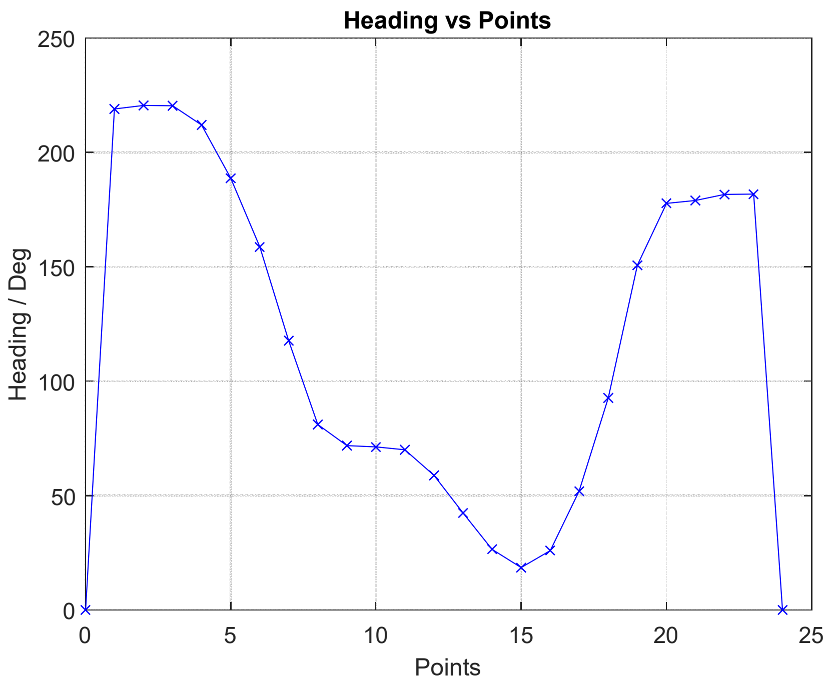


Figure 59. Heading of Points on Test Route

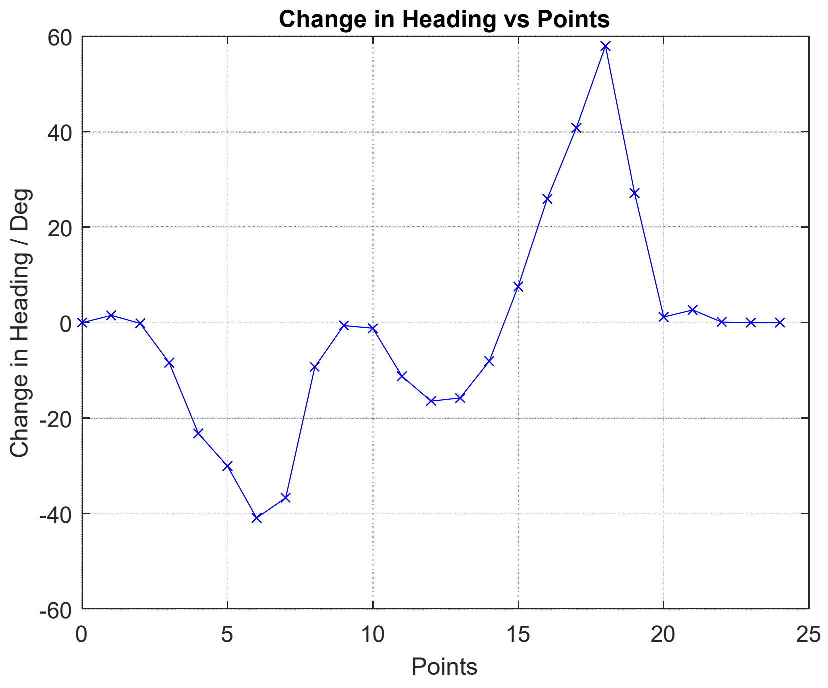


Figure 60. Change of Heading of Points on Test Route

With the positive results seen from the test, we are able to proceed to compute the change in heading for all the points on the actual route. The newly computed data for change in heading then allows us to carry out statistical analysis to calculate the standard deviation of the distribution. Using the same example of the Carmel Valley road, we find the standard deviation of the change in heading was calculated to be 5.643 degrees, which classifies the route as a type 3 road. This is close to the results obtained by Driels [10], as discussed in Chapter II, and helps to validate the approach adopted in this study. Having validated the approach by obtaining similar results with previous works, we then proceed to apply the program to routes in other countries where mapping of routes may be less extensive than that in the United States. The program is used for selected routes in North Korea, Syria, and Afghanistan. The program works well, and the results are presented in Figures 61 to Figure 63, respectively.

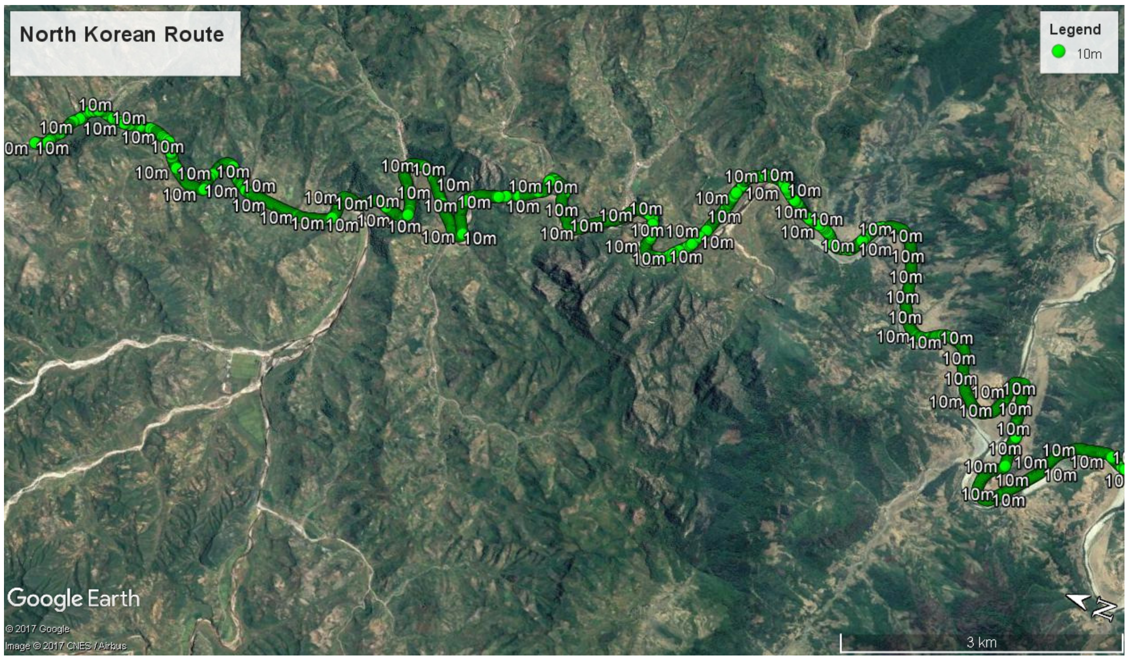


Figure 61. Processed North Korean Route—Type 2

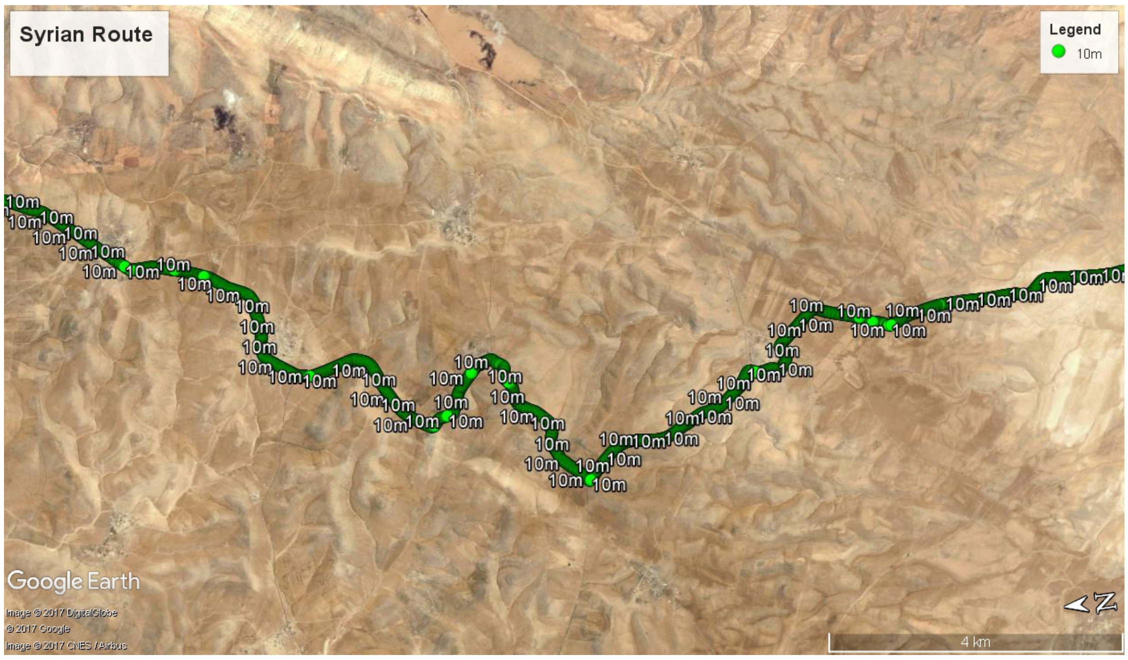


Figure 62. Processed Syrian Route—Type 2

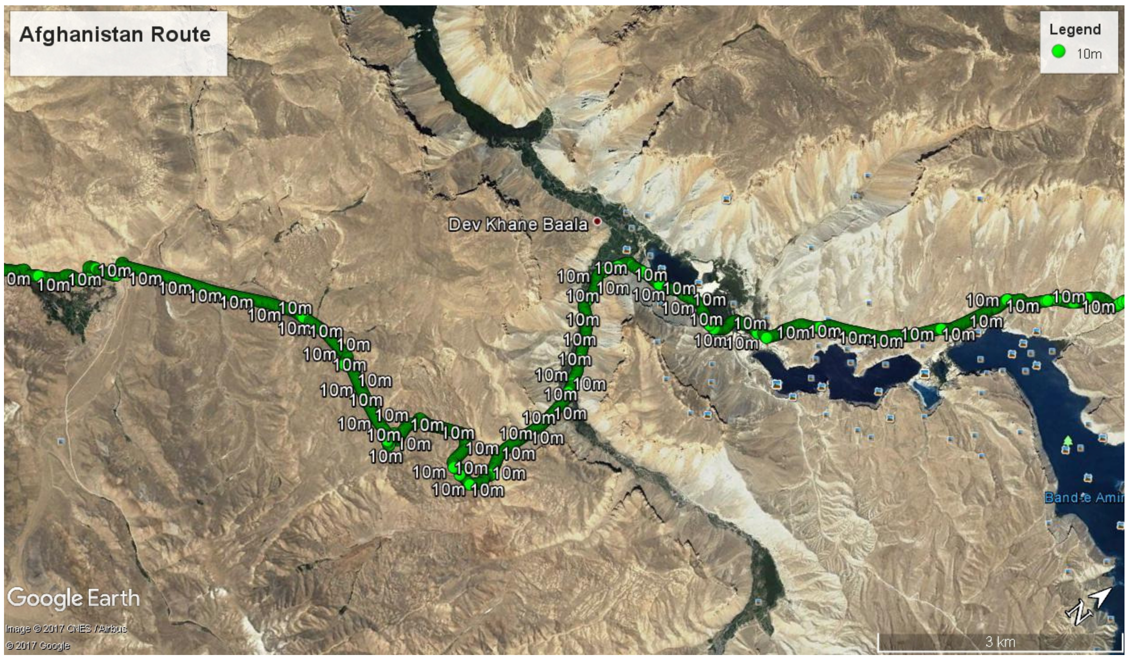
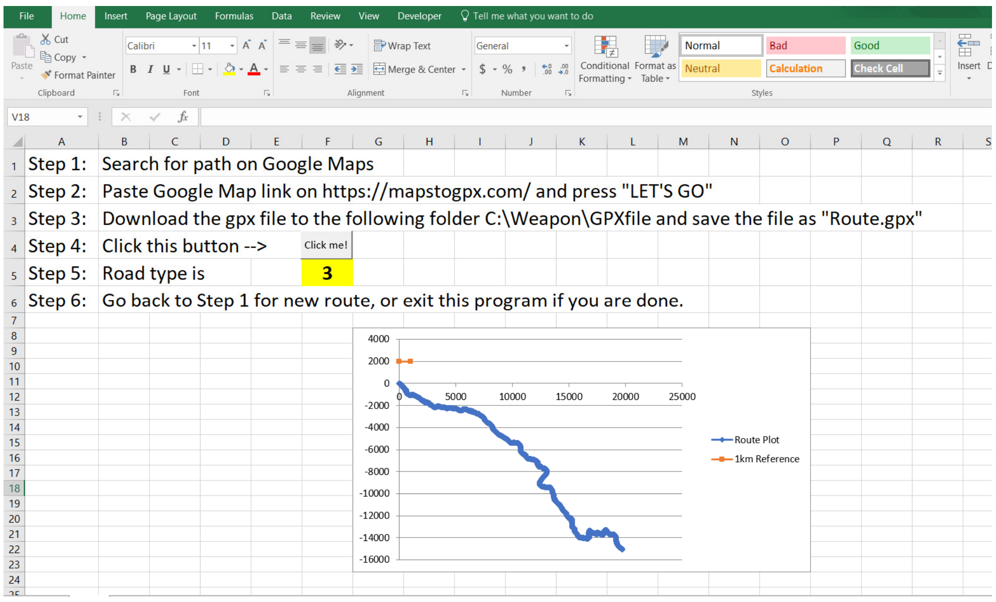


Figure 63. Processed Afghanistan Route—Type 3

From the results for the routes shown, it can be seen that the approach works for the routes that can be selected using Google Maps and imported into the software tool.

The model is initially developed and tested in MATLAB. As discussed earlier, a fresh copy is translated into the VBA environment to suit the intended user. With the click of a single macro button by the user, the VBA program will automatically extract the route data from the GPX file and analyze the route to compute the standard deviation in the change of heading, thereby categorizing the route into the appropriate road type based on the results obtained. The user interface in the VBA environment is presented in Figure 64.



Step 1:	Search for path on Google Maps
Step 2:	Paste Google Map link on https://mapstogpx.com/ and press "LET'S GO"
Step 3:	Download the gpx file to the following folder C:\Weapon\GPXfile and save the file as "Route.gpx"
Step 4:	Click this button --> Click me!
Step 5:	Road type is 3
Step 6:	Go back to Step 1 for new route, or exit this program if you are done.

Figure 64. Route Analysis in VBA Environment

Both models are able to produce the same results, verifying that the working approach is successfully implemented in an interface suitable for the intended users.

A copy of the code for the route analysis implemented for MATLAB and VBA can be found in Appendix F and Appendix G, respectively.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. CONCLUSION AND RECOMMENDATIONS

A. CONCLUSION

This thesis has achieved the objectives set by successfully generating the TSPI files for different synthesized routes that are representative of actual target routes of interest. With these TSPI files obtained, the routes can then be fed into the weapon IFS model for weapon assessment against a moving target traveling along the specific route type. The results of the weapon assessment can then be compiled to create a quick look-up reference for assessment of the effectiveness of a specific weapon against moving targets moving along route of a particular road type.

A program, written in the VBA environment, is also developed to enable quick categorization of an actual road of interest into one of the road profiles described, and this program enables the weaponeer to obtain quick estimates of the effectiveness of various weapon types available and facilitates a quick selection of weapon type for optimal probability of kill.

The key products of this thesis include:

- the MATLAB program for generation of synthesized constrained routes;
- the MATLAB program for generating TSPI files for synthesized routes.;
- the Simulink program for generating expected unconstrained routes for sea vessels;
- the MATLAB program for generating TSPI files for sea routes;
- the MATLAB program for digitization of map routes and categorization of map routes into specific road types; and

- the VBA program for digitization of map routes and categorization of map routes into specific road types.

B. RECOMMENDATIONS

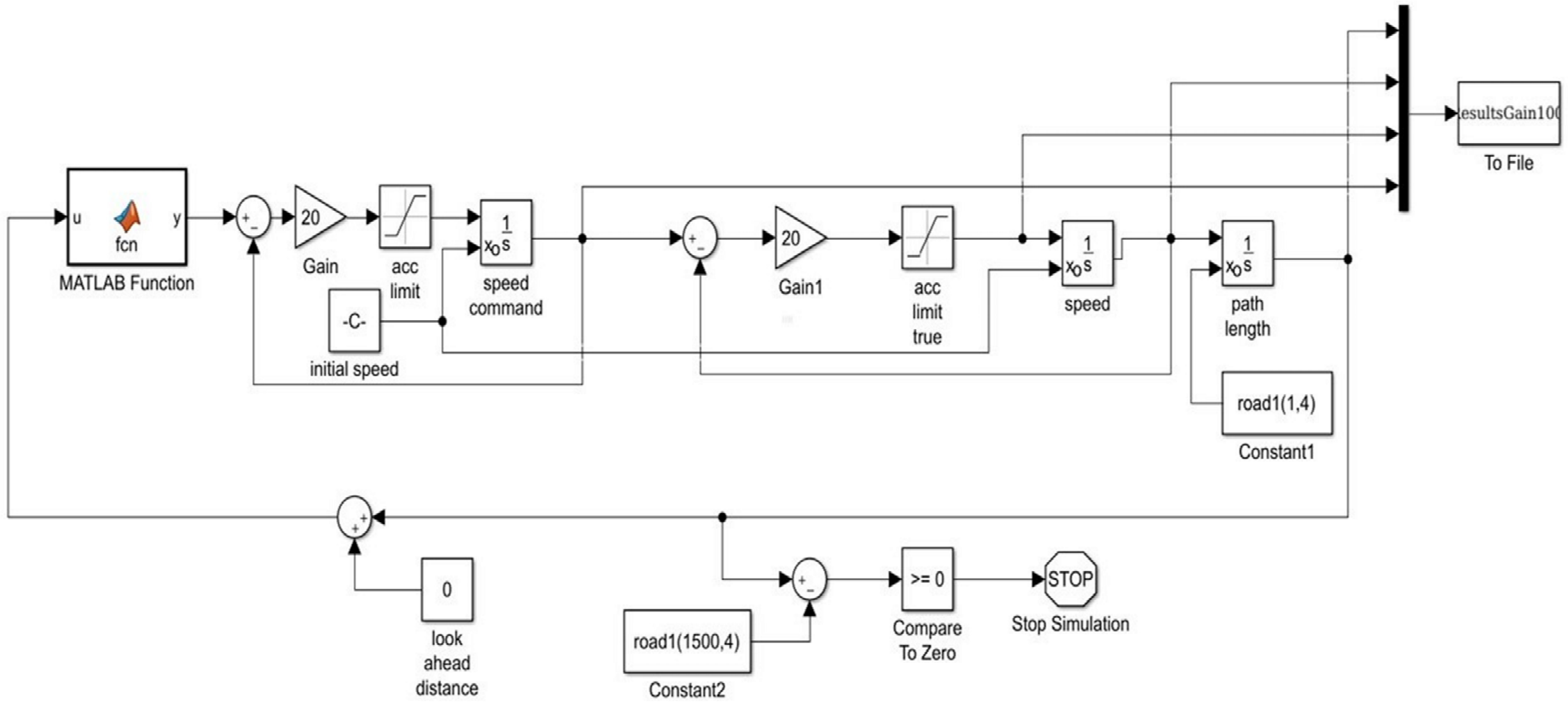
This thesis can be further extended into the following areas for future works:

- The velocity computation can be done using the Simulink model discussed earlier to better capture the effects of vehicle dynamics to produce results that are more representative of the actual vehicle performance.
- The velocity computation can be optimized to determine the optimal time for a given vehicle to travel a route in minimal time.
- The TSPI data obtained can be compared to actual vehicle trials to compare the theoretical and empirical results to verify the approach and assumptions used.

APPENDIX A. MATLAB CODE FOR SYNTHESIZED ROUTE GENERATION

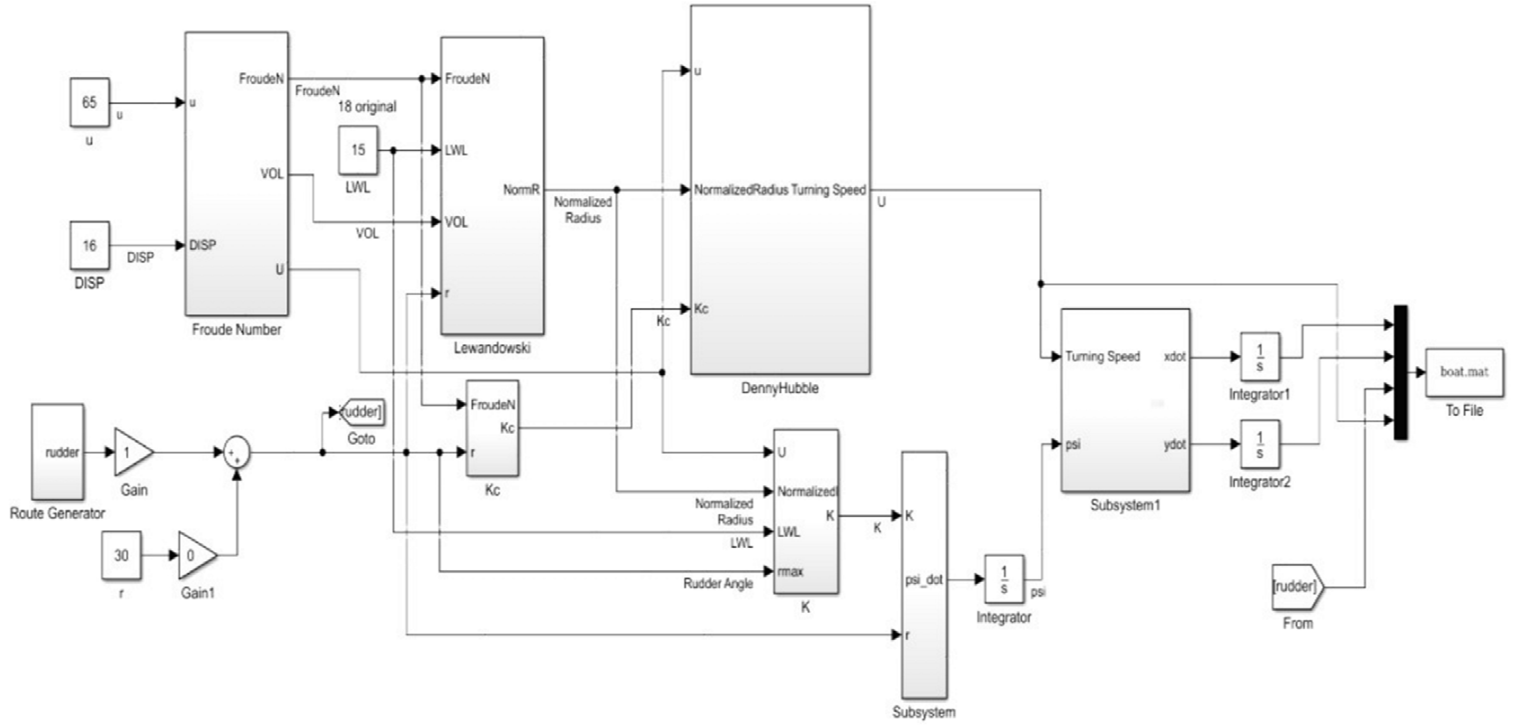
```
clear all
clc

heading_limit=2;          % # of sigmas heading is
limited

%-------- New data set for road types 1-6 ---------
 mean_heading=  [135 135 135 135 135 135];
 sigma_heading= [25 29 41 53 66 73];
 mean_dheading= [0 0 0 0 0 0];
 sigma_dheading= [0.5 2.5 7 12 17 25];
 total_km =   [40 40 40 40 40 40];
 road_type=menu(' Choose a road type ','Type 1',' Type
2 ',....
' Type 3 ',' Type 4 ','Type 5',' Type 6 ');
 r_type=['Type 1';'Type 2';'Type 3';'Type 4';'Type
5';'Type 6'];
 r_type_data=cellstr(r_type);

% get statistics for the chosen road type:
av_heading = 90 - mean_heading(road_type);  % rotate
coordinates from north to east
sig_heading=sigma_heading(road_type);
dhmean = mean_dheading(road_type);
dhstd_dev = sigma_dheading(road_type);
h_limit = heading_limit*sig_heading;     % heading
constraint limit


heading(1) = av_heading;  % starting heading
seg_dist = 10;    % 10 meter segments
npts = total_km(road_type) * 1000 / seg_dist;


% Randomly sample from the normal distribution
% Each r value gives you the change in heading for the
next 10 m of road
for i=1:npts
  %r(i) = random('norm',dhmean,dhstd_dev);
  r(i)=randn*dhstd_dev+dhmean;
end
stdev_data=std(r)     % Calculate the actual sigma of
this data set

%
% Here's the road building part, which turns a heading
change into x and y
% coordinates starting at easting(1), northing(1).
easting(1) = 615788;
northing(1) = 4035674;

for i=2:npts+1

  heading(i) = heading(i-1) + r(i-1);

  %try to keep road from making circles:
  if heading(i) > (av_heading + h_limit) | heading(i) <
(av_heading - h_limit)
    heading(i) = heading(i-1) - r(i-1);
  end
  easting(i) = easting(i-1) + seg_dist*cos(heading(i-
1)*pi/180);
  northing(i) = northing(i-1) + seg_dist*sin(heading(i-
1)*pi/180);
end

% plot the road
figure(2)
plot(easting/1000, northing/1000,'LineWidth',1.25)
axis('equal'); grid
% title('Synthesized Road')
title(r_type_data(road_type))
xlabel('Easting (km)')
ylabel('Northing (km)')
```

APPENDIX B. SIMULINK MODEL FOR VELOCITY COMPUTATION



THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. SIMULINK MODEL FOR SEA ROUTE GENERATION



THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. MATLAB CODE FOR SYNTHESIZED ROUTE PROCESSING INTO TSPI FORMAT

```
clear all
clc

load('Type3.mat')

for i=1:length(type3)
  newroute(i,1) = type3(i,1) - type3(1,1); % initialise
1st data point as origin
  newroute(i,2) = type3(i,2) - type3(1,2);
end

%% Initialization for Computing Speed
veh_limit = 60; % vehicle speed limit in mph
m2ft = 3.28084; % conversion from metres to feet
mph2ms = 1.60934 * 1000 / 3600; % conversion from mph
to m/s
coeff_friction = 0.7; % coefficient of friction
safety_factor = 1; % speed safety factor
interval = 10; % distance between subsequent points is
10m
check = 10; % max number of points to check prior and
after
decel_limit = 0.6 * 9.81; % deceleration limit, from g
to m/s2
accel_limit = 0.3 * 9.81; % acceleration limit, from g
to m/s2

newroute(1,3) = 0; % initialize distance
newroute(1,4:9) = 0;
newroute(1,10) = safety_factor * min(sqrt(15 * 1e15 *
m2ft) * (coeff_friction + 0.01 * 0),veh_limit) *
mph2ms; % initialize vehicle at speed limit, with bank
angle = 0

for i=2:length(newroute(:,1))-1
  newroute(i,3) = ((newroute(i,1)-newroute(i-1,1)).^2 +
(newroute(i,2)-newroute(i-1,2)).^2) .^ 0.5; % there are
values less than zero as previous computation was based
on accumulated distance travelled
```

```
  newroute(i,4) = (newroute(i,2) - newroute(i-1,2)) /
(newroute(i,1) - newroute(i-1,1)); % gradient between
point i and i-1
  newroute(i,5) = (newroute(i+1,2) - newroute(i,2)) /
(newroute(i+1,1) - newroute(i,1)); % gradient between
point i and i+1
  newroute(i,6) = (newroute(i,4) * newroute(i,5) *
(newroute(i+1,2) - newroute(i-1,2)) + newroute(i,4) *
(newroute(i,1) + newroute(i+1,1)) - newroute(i,5) *
(newroute(i,1) + newroute(i-1,1))) / (2 *
(newroute(i,4) - newroute(i,5))); % compute x-
coordinate of centre of circle of curvature
  newroute(i,7) = - 1 / newroute(i,5) * (newroute(i,6)
- (newroute(i,1) + newroute(i+1,1)) / 2) +
(newroute(i,2) + newroute(i+1,2)) / 2; % compute y-
coordinate of centre of circle of curvature
  newroute(i,8) = min(((newroute(i,6) - newroute(i,1))
^ 2 + (newroute(i,7) - newroute(i,2)) ^ 2) ^ 0.5,1e15);
% radius of curvature in metres, 1e15 for those with
infinite values, i.e collinear points
  newroute(i,9) = 0; % road bank angle
  newroute(i,10) = safety_factor * min(sqrt(15 *
(newroute(i,8) * m2ft) * (coeff_friction + 0.01 *
newroute(i,9))),veh_limit) * mph2ms; % max speed limit
at point i in m/s, converted from mph using radius of
curvature in feet
  newroute(i,16) = sum(newroute(1:i,3)); % Displacement
end

%% Generate Speeds at points
newroute(1,11) = sqrt(newroute(2,10)^2 + 2 *
decel_limit * interval);
newroute(1,12) = sqrt(newroute(2,10)^2 - 2 *
accel_limit * interval);
newroute(1,13) = max(newroute(1,11),newroute(1,12));

for i=2:length(newroute(:,1))-1
  % check for deceleration constraint
  acc_check1 = NaN(check,1);
  vel_check1 = NaN(check,1);
  for j=1:min((length(newroute)-1) - i, check) %
iteration for check on speed limits on points ahead
    if newroute(i,10) > newroute(i+j,10)
```

```
      vel_check1(j) = sqrt(newroute(i+j,10)^2 + 2 * j *
decel_limit * interval);
    else
    end
  end
  newroute(i,11) = min(min(vel_check1),newroute(i,10));

  % check for acceleration constraint
  acc_check2 = NaN(check,1);
  vel_check2 = NaN(check,1);

  for j=1:min(i-1, check) % iteration for check on
speed limits on points prior
    if newroute(i,10) > newroute(i-j,10)
      vel_check2(j) = sqrt(newroute(i-j,10)^2 + 2 * j *
accel_limit * interval);
    else
    end
  end
  newroute(i,12) = min(min(vel_check2),newroute(i,10));
  newroute(i,13) = min(newroute(i,11),newroute(i,12));
% Actual speed/ highest speed possible
end

%% Generate time between points
for i=1:length(newroute(:,1))-2
  newroute(i,14) = (newroute(i+1,13)^2 -
newroute(i,13)^2) / (2 * interval); % Acceleration
  newroute(i,15) = 2 * interval / (newroute(i,13) +
newroute(i+1,13)); % time from point i and i+1
end

%% Generate Points at 1 sec Intervals
carryover = 0;
k = 2;
time = newroute(:,15);
newroutetime(1,1) = newroute(1,1);
newroutetime(1,2) = newroute(1,2);
newroutetime(1,4) = 0;
for j=1:length(time)-1
  if time(j) + carryover < 1
    carryover = time (j) + carryover;
  else
```

```
    for count=1:floor((carryover + time(j))/1)
      newroutetime(k,8)  = k - 1; % total time
      newroutetime(k,1)=newroute(j,1)+(newroute(j+1,1)-
newroute(j,1))*(count-carryover)/time(j);
      newroutetime(k,2)=newroute(j,2)+(newroute(j+1,2)-
newroute(j,2))*(count-carryover)/time(j);

newroutetime(k,9)=newroute(j,13)+(newroute(j+1,13)-
newroute(j,13))*(count-carryover)/time(j);% speed at
point k
      k=k+1;
    end
    carryover = carryover + time(j) - count;
  end
end

%% Heading and Speed for Time-Based route
for i=2:length(newroutetime)-1 % heading
  newroutetime(i,4)  = newroutetime(i+1,1) -
newroutetime(i-1,1); % delta x
  newroutetime(i,5)  = newroutetime(i+1,2) -
newroutetime(i-1,2); % delta y
  newroutetime(i,6)  =
atan2(newroutetime(i,5),newroutetime(i,4)); % heading
with horizontal as 0 rad
  if newroutetime(i,6) > pi/2 && newroutedist(i,6) < pi
% heading with vertical as 0 rad
    newroutetime(i,7)  = 2 * pi + ( pi / 2 -
newroutetime(i,6));
  else
    newroutetime(i,7)  = pi / 2 - newroutetime(i,6);
  end
  newroutetime(i,10)  = newroutetime(i,9)  *
sin(newroutetime(i,7));
  newroutetime(i,11)  = newroutetime(i,9)  *
cos(newroutetime(i,7));
end

%% Conversion From ENU to ECEF to Geodetic
a = 6378137.0; % WGS-84 Earth semimajor axis (m)
b = 6356752.3142; % WGS-84 Earth semiminor axis (m)
f = (a-b)/a; % Ellipsoid Flatness
e_sq = f * (2 - f); % Square of Eccentricity
```

```
epsilon = e_sq / (1 - e_sq);
h = 0;

route1(1,1) = 0; % initialise synthesized route start
point lat/lon to 0,0
route1(1,2) = 0;
route1(1,3) = sin(route1(1,1)); % sin (lat)
route1(1,4) = a / sqrt(1-e_sq * route1(1,3) *
route1(1,3)); % N, distance from the surface to the z-
axis along the ellipsoid normal
route1(1,5) = (h + route1(1,4)) * cos(route1(1,1)) *
cos(route1(1,2)); % converting to ECEF x-coordinates
route1(1,6) = (h + route1(1,4)) * cos(route1(1,1)) *
sin(route1(1,2)); % converting to ECEF y-coordinates
route1(1,7) = (h + (1 - e_sq) * route1(1,4)) *
sin(route1(1,1)); % converting to ECEF z-coordinates
for i=1:length(newroutetime) % converting points for 1s
intervals back to Lat/Lon
  route3(i,18) = -sin(route1(1,2)) * newroutetime(i,1)
- sin(route1(1,1)) * cos(route1(1,2)) *
newroutetime(i,2) + cos(route1(1,1)) * cos(route1(1,2))
* newroutetime(i,3) + route1(1,5); % convert back to
ECEF x-coordinate
  route3(i,19) = cos(route1(1,2)) * newroutetime(i,1) -
sin(route1(1,1)) * sin(route1(1,2)) * newroutetime(i,2)
+ cos(route1(1,1)) * sin(route1(1,2)) *
newroutetime(i,3) + route1(1,6);% convert back to ECEF
y-coordinate
  route3(i,20) = cos(route1(1,1)) * newroutetime(i,2) +
sin(route1(1,1)) * newroutetime(i,3) + route1(1,7);%
convert back to ECEF z-coordinate
  route3(i,21) = sqrt(route3(i,18)^2 + route3(i,19)^2);
% p
  route3(i,22) = atan((route3(i,20) * a)/ (route3(i,21)
* b)); % q
  route3(i,23) = atan2((route3(i,20) + epsilon * b *
(sin(route3(i,22)))^3) , (route3(i,21) - e_sq * a *
(cos(route3(i,22)))^3)); % Lat
  route3(i,24) = atan2 (route3(i,19),route3(i,18)); %
Lon
end
```

```
NRTLatLon(:,1)=route3(:,23)*180/pi; %converting 1s x-
coordinates back to Lon
NRTLatLon(:,2)=route3(:,24)*180/pi; %converting 1s y-
coordinates back to Lat
%% Extracting TSPI
TSPI = [newroutetime(:,8) NRTLatLon(:,1) NRTLatLon(:,2)
newroutetime(:,10) newroutetime(:,11)];
```

APPENDIX E. MATLAB CODE FOR SEA ROUTE PROCESSING

```
clear all
clc

%% Simulate the Simulink Model
sim('Sea_Vessel_Plant.slx',90); % runnning Model in
Simulink
%% Read the timeseries object from the data file
load boat.mat
%% Parcing
time_sim=boat_results.time;
route(:,1)=boat_results.Data(:,1);
route(:,2)=boat_results.Data(:,2);
rudder = boat_results.Data(:,3);
speed = boat_results.Data(:,4);

%% Convert Intervals of 10m
for i=1:length(route(:,1))-1 %finding difference in x
and y for points i and i+1
diff1x(i)=route(i+1,1)-route(i,1);
diff1y(i)=route(i+1,2)-route(i,2);
end
k=2; %initiatise k=2
carryover(1)=0; % initiatise carryover=0
addx=0; addy=0;
newroute=[route(1,1) route(1,2)];

for j=1:i
  distance(j)=(diff1x(j)^2+diff1y(j)^2)^0.5; %computing
distance between points j and j+1
  if carryover(j) > 0
    if (carryover(j)+distance(j))>=10
      for count=1:floor((carryover(j)+distance(j))/10)
        if count==1
          newroute(k,1)=newroute(k-
1,1)+diff1x(j)/distance(j)*(10-carryover(j))+addx;
          newroute(k,2)=newroute(k-
1,2)+diff1y(j)/distance(j)*(10-carryover(j))+addy;
          k=k+1;
        else
```

```
          newroute(k,1)=newroute(k-
1,1)+diff1x(j)/distance(j)*10;
          newroute(k,2)=newroute(k-
1,2)+diff1y(j)/distance(j)*10;
          k=k+1;
        end
      end
      carryover(j+1)=distance(j)-(10-carryover(j))-
(count-1)*10;
      addx=diff1x(j)/distance(j)*carryover(j+1);
      addy=diff1y(j)/distance(j)*carryover(j+1);
    else
      carryover(j+1)=carryover(j)+distance(j);
      addx=addx+diff1x(j);
      addy=addy+diff1y(j);
    end
  else
    if distance(j)>=10 % check if distance to next
point is > 10m
      for count=1:floor(distance(j)/10)
        newroute(k,1)=newroute(k-
1,1)+diff1x(j)/distance(j)*10;
        newroute(k,2)=newroute(k-
1,2)+diff1y(j)/distance(j)*10;
        k=k+1;
      end
      carryover(j+1)=distance(j)-(count)*10;
      addx=diff1x(j)/distance(j)*carryover(j+1);
      addy=diff1y(j)/distance(j)*carryover(j+1);
    elseif distance(j)<10 % check if distance to next
point is < 10m
      carryover(j+1)=distance(j);
      addx=diff1x(j);
      addy=diff1y(j);
    end
  end
end

%% Speed
for i=1:length(newroute(:,1))
  row_index = find(newroute(i,1)<route(:,1),1)-1;
  if row_index < length(newroute(:,1))
  newroute(i,3) = (newroute(i,1) - route(row_index,1))
/ (route(row_index+1,1) - route(row_index,1)) *
(speed(row_index + 1) - speed(row_index)) +
speed(row_index); % interpolate speed
  end
end

%% Generate time between points
interval = 10;
for i=1:length(newroute(:,1))-1
  newroute(i,4) = 2 * interval / (newroute(i,3) +
newroute(i+1,3)); % time from point i and i+1
end

%% Generate Points at 1 sec Intervals
carryover = 0;
k = 2;
time = newroute(:,4);
newroutetime(1,1) = time_sim(1);
newroutetime(1,2) = newroute(1,1);
newroutetime(1,3) = newroute(1,2);
for j=1:length(time)-1
  if time(j) + carryover < 1
    carryover = time (j) + carryover;
  else
    for count=1:floor((carryover + time(j))/1)
      newroutetime(k,1) = k - 1;
      newroutetime(k,2)=newroute(j,1)+(newroute(j+1,1)-
newroute(j,1))*(count-carryover)/time(j);
      newroutetime(k,3)=newroute(j,2)+(newroute(j+1,2)-
newroute(j,2))*(count-carryover)/time(j);
      k=k+1;
    end
    carryover = carryover + time(j) - count;
  end
end
%% Speed
for i=1:length(newroutetime(:,1))
  row_index = find(newroutetime(i,2)<route(:,1),1)-1;
  if row_index < length(route(:,1))
  newroutetime(i,4) = (newroutetime(i,2) -
route(row_index,1)) / (route(row_index+1,1) -
route(row_index,1)) * (speed(row_index + 1) -
speed(row_index)) + speed(row_index); % interpolate
speed
  end
end


%% Angle
newroutetime(1,8) = pi / 2;
newroutetime(1,9) = newroutetime(1,7) / pi * 180;
for i=2:length(newroutetime(:,1))-1
  newroutetime(i,5) = newroutetime(i+1,2) -
newroutetime(i-1,2); % delta x
  newroutetime(i,6) = newroutetime(i+1,3) -
newroutetime(i-1,3); % delta y
  newroutetime(i,7) =
atan2(newroutetime(i,6),newroutetime(i,5)); % heading
with horizontal as 0 rad
  if newroutetime(i,7) > pi/2 && newroutetime(i,7) < pi
% heading with vertical as 0 rad
    newroutetime(i,8) = 2 * pi + ( pi / 2 -
newroutetime(i,7));
  else
    newroutetime(i,8) = pi / 2 - newroutetime(i,7);
  end
  newroutetime(i,9) = newroutetime(i,8) / pi * 180;
end

%% Rudder

for i=1:length(newroutetime(:,1))
  row_index = find(newroutetime(i,2)<route(:,1),1)-1;
  if row_index < length(route(:,1))
  newroutetime(i,10) = (newroutetime(i,2) -
route(row_index,1)) / (route(row_index+1,1) -
route(row_index,1)) * (rudder(row_index + 1) -
rudder(row_index)) + rudder(row_index); % interpolate
speed
  end
end

for i=1:length(newroute(:,1))
  row_index = find(newroute(i,1)<route(:,1),1)-1;
  if row_index < length(route(:,1))
  newroute(i,5) = (newroute(i,1) - route(row_index,1))
/ (route(row_index+1,1) - route(row_index,1)) *
(rudder(row_index + 1) - rudder(row_index)) +
rudder(row_index); % interpolate speed
  end
end

TSPI = [newroutetime(:,1) newroutetime(:,2)
newroutetime(:,3) newroutetime(:,4).*
sin(newroutetime(:,8)) newroutetime(:,4).*
cos(newroutetime(:,8))];
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX F. MATLAB CODE FOR ROUTE ANALYSIS

```
clear all
clc

%% Generate X Y Z from GPX

route = gpxread('Carmel_Valley.gpx', 'FeatureType',
'track');

%% Defining WGS-84 Geodetic Constants
a = 6378137.0; % WGS-84 Earth semimajor axis (m)
b = 6356752.3142; % WGS-84 Earth semiminor axis (m)
f = (a-b)/a; % Ellipsoid Flatness
e_sq = f * (2 - f); % Square of Eccentricity
epsilon = e_sq / (1 - e_sq);
h = 0;

route1Lat=route.Latitude'; % extracting latitude, phi
route1Lon=route.Longitude'; % extracting longitude,
lambda
route1Lat2rad=route.Latitude'.*(pi/180); % extracting
latitude, phi
route1Lon2rad=route.Longitude'.*(pi/180); % extracting
longitude, lambda
route1=[route1Lat2rad route1Lon2rad];
for i=1:length(route1)
  route1(i,3) = sin(route1(i,1)); % sin (lat)
  route1(i,4) = a / sqrt(1-e_sq * route1(i,3) *
route1(i,3)); % N, distance from the surface to the z-
axis along the ellipsoid normal
  route1(i,5) = (h + route1(i,4)) * cos(route1(i,1)) *
cos(route1(i,2)); % converting to ECEF x-coordinates
  route1(i,6) = (h + route1(i,4)) * cos(route1(i,1)) *
sin(route1(i,2)); % converting to ECEF y-coordinates
  route1(i,7) = (h + (1 - e_sq) * route1(i,4)) *
sin(route1(i,1)); % converting to ECEF z-coordinates
  route1(i,8) = route1(i,5) - route1(1,5); % delta x
  route1(i,9) = route1(i,6) - route1(1,6); % delta y
  route1(i,10) = route1(i,7) - route1(1,7); % delta z
  route1(i,11) = -sin(route1(1,2)) * route1(i,8) +
cos(route1(1,2)) * route1(i,9); % Easting
```

```
  route1(i,12) = -cos(route1(1,2)) * sin(route1(1,1)) *
route1(i,8) - sin(route1(1,1)) * sin(route1(1,2)) *
route1(i,9) + cos(route1(1,1)) * route1(i,10); %
Northing
  route1(i,13) = cos(route1(1,1)) * cos(route1(1,2)) *
route1(i,8) + cos(route1(1,1)) * sin(route1(1,2)) *
route1(i,9) + sin(route1(1,1)) * route1(i,10); % Up
end

%% Convert Intervals of 10m
for i=1:find(route1(:,11)==route1(end,11))-1 %finding
difference in x and y for points i and i+1
diff1x(i)=route1(i+1,11)-route1(i,11);
diff1y(i)=route1(i+1,12)-route1(i,12);
end
k=2; %initiatise k=2
carryover(1)=0; % initiatise carryover=0
addx=0; addy=0;
newroute=[route1(1,11) route1(1,12)];

for j=1:i
  distance(j)=(diff1x(j)^2+diff1y(j)^2)^0.5; %computing
distance between points j and j+1
  if carryover(j) > 0
    if (carryover(j)+distance(j))>=10
      for count=1:floor((carryover(j)+distance(j))/10)
        if count==1
          newroute(k,1)=newroute(k-
1,1)+diff1x(j)/distance(j)*(10-carryover(j))+addx;
          newroute(k,2)=newroute(k-
1,2)+diff1y(j)/distance(j)*(10-carryover(j))+addy;
          k=k+1;
        else
          newroute(k,1)=newroute(k-
1,1)+diff1x(j)/distance(j)*10;
          newroute(k,2)=newroute(k-
1,2)+diff1y(j)/distance(j)*10;
          k=k+1;
        end
      end
      carryover(j+1)=distance(j)-(10-carryover(j))-
(count-1)*10;
      addx=diff1x(j)/distance(j)*carryover(j+1);
      addy=diff1y(j)/distance(j)*carryover(j+1);
    else
      carryover(j+1)=carryover(j)+distance(j);
      addx=addx+diff1x(j);
      addy=addy+diff1y(j);
    end
  else
    if distance(j)>=10 % check if distance to next
point is > 10m
      for count=1:floor(distance(j)/10)
        newroute(k,1)=newroute(k-
1,1)+diff1x(j)/distance(j)*10;
        newroute(k,2)=newroute(k-
1,2)+diff1y(j)/distance(j)*10;
        k=k+1;
      end
      carryover(j+1)=distance(j)-(count)*10;
      addx=diff1x(j)/distance(j)*carryover(j+1);
      addy=diff1y(j)/distance(j)*carryover(j+1);
    elseif distance(j)<10 % check if distance to next
point is < 10m
      carryover(j+1)=distance(j);
      addx=diff1x(j);
      addy=diff1y(j);
    end
  end
end


%% Computing UP Coordinates from Northing

rindex=0;
newroutedist(:,1) = newroute(:,1);
newroutedist(:,2) = newroute(:,2);
for i=1:length(newroutedist) % Computing UP Coordinates
from Northing for 10m interval
  rindex=find(newroutedist(i,2)>=route1(:,12),1);
  newroutedist(i,3)=(newroutedist(i,2)-
route1(rindex,12))/(route1(rindex+1,12)-
route1(rindex,12)) * (route1(rindex+1,13)-
route1(rindex,13)) + route1(rindex,13);
end
```

```
%% Heading Computation for Distance-Based

for i=2:length(newroutedist)-1
  newroutedist(i,4) = newroutedist(i+1,1) -
newroutedist(i-1,1); % delta x
  newroutedist(i,5) = newroutedist(i+1,2) -
newroutedist(i-1,2); % delta y
  newroutedist(i,6) =
atan2(newroutedist(i,5),newroutedist(i,4)); % heading
with horizontal as 0 rad
  if newroutedist(i,6) > pi/2 && newroutedist(i,6) < pi
% heading with vertical as 0 rad
    newroutedist(i,7) = 2 * pi + ( pi / 2 -
newroutedist(i,6));
  else
    newroutedist(i,7) = pi / 2 - newroutedist(i,6);
  end
end

for i=2:length(newroutedist)-2
  if newroutedist(i+1,7) - newroutedist(i,7) >= pi &&
newroutedist(i+1,7) - newroutedist(i,7) <= 3 * pi %
change in heading
    newroutedist(i,8) = newroutedist(i+1,7) -
newroutedist(i,7) - 2 * pi;
  elseif newroutedist(i+1,7) - newroutedist(i,7) <= -pi
&& newroutedist(i+1,7) - newroutedist(i,7) >= -3 * pi
    newroutedist(i,8) = newroutedist(i+1,7) -
newroutedist(i,7) + 2 * pi;
  else
    newroutedist(i,8) = newroutedist(i+1,7) -
newroutedist(i,7);
  end
end

if std(newroutedist(2:length(newroutedist)-2,8))/pi*180
>= 0 && std(newroutedist(2:length(newroutedist)-
2,8))/pi*180 <= 1 % classify road type
  roadtype = 1;
elseif std(newroutedist(2:length(newroutedist)-
2,8))/pi*180 > 1 &&
std(newroutedist(2:length(newroutedist)-2,8))/pi*180 <=
4.5
```

```
  roadtype = 2;
elseif std(newroutedist(2:length(newroutedist)-
2,8))/pi*180 > 4.5 &&
std(newroutedist(2:length(newroutedist)-2,8))/pi*180 <=
9.5
  roadtype = 3;
elseif std(newroutedist(2:length(newroutedist)-
2,8))/pi*180 > 9.5 &&
std(newroutedist(2:length(newroutedist)-2,8))/pi*180 <=
14.5
  roadtype = 4;
elseif std(newroutedist(2:length(newroutedist)-
2,8))/pi*180 > 14.5 &&
std(newroutedist(2:length(newroutedist)-2,8))/pi*180 <=
19.5
  roadtype = 5;
elseif std(newroutedist(2:length(newroutedist)-
2,8))/pi*180 > 19.5 &&
std(newroutedist(2:length(newroutedist)-2,8))/pi*180 >=
30.5
  roadtype = 6;
else
  roadtype = 7;
end

fprintf('\nThe sigma change in heading is %g degrees,
therefore it is road type %d.\n',
std(newroutedist(2:length(newroutedist)-2,8))/pi*180,
roadtype)
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX G. VBA CODE FOR ROUTE ANALYSIS

```
Sub Main()
With ThisWorkbook
  .Sheets("Workspace").Visible = True
  Call ImportGPX
  Call CopyColumnToWorkbook
  Call FillDownXY
  Call convertm
  Call HeadingXY
  Call Chart
  Call RemoveGPX
  '.Sheets("Workspace").Visible = xlSheetVeryHidden
End With
End Sub

Sub ImportGPX() ' Import GPX data into worksheet

With ThisWorkbook
    .Sheets.Add(After:=.Sheets(.Sheets.Count)).Name = "GPX data"
End With

Application.DisplayAlerts = False
ActiveWorkbook.XmlImport                    Url:="C:\Weapon\GPXfile\Route.gpx",
ImportMap:=Nothing, _
  Overwrite:=True, Destination:=Worksheets("GPX data").Range("$A$1")
Application.DisplayAlerts = True

End Sub
Sub CopyColumnToWorkbook()
  Dim r1 As Range
With ThisWorkbook
  .Sheets("GPX data").Select
  Set r1 = Range("L:M")
  .Sheets("Workspace").Select
  .Sheets("Workspace").Range("A4:Z1000000").ClearContents
  r1.Copy Range("A1")
End With
End Sub


Sub FillDownXY()

  Dim strFormulas1(1 To 2), strFormulas2(1 To 3) As Variant
  Dim LastRow As Long
  Dim a, b, f, e_sq, epsilon, h As Long

  a = 6378137         'WGS-84 Earth semimajor axis (m)
  b = 6356752.3142      'WGS-84 Earth semiminor axis (m)
  f = (a - b) / a       'Ellipsoid Flatness
  e_sq = f * (2 - f)     'Square of Eccentricity
  epsilon = e_sq / (1 - e_sq)

  h = 0

  LastRow = Range("A" & Rows.Count).End(xlUp).Row

  With ThisWorkbook.Sheets("Workspace")
    strFormulas1(1) = "=A4/180*pi()"
    strFormulas1(2) = "=B4/180*pi()"

    .Range("C4:D4").Formula = strFormulas1
    .Range("C4:D" & LastRow).FillDown

    For j = 1 To LastRow - 3
      .Range("E" & j + 3).Value = a / ((1 - e_sq * Sin(.Range("C" & j + 3).Value) *
Sin(.Range("C" & j + 3).Value)) ^ 0.5) 'N, distance from the surface to the z-axis
along the ellipsoid normal
      .Range("F" & j + 3).Value = (h + .Range("E" & j + 3).Value) * Cos(.Range("C"
& j + 3).Value) * Cos(.Range("D" & j + 3).Value) 'converting to ECEF x-
coordinates
      .Range("G" & j + 3).Value = (h + .Range("E" & j + 3).Value) * Cos(.Range("C"
& j + 3).Value) * Sin(.Range("D" & j + 3).Value) 'converting to ECEF y-
coordinates
      .Range("H" & j + 3).Value = (h + (1 - e_sq) * .Range("E" & j + 3).Value) *
Sin(.Range("C" & j + 3).Value) 'converting to ECEF z-coordinates
      .Range("I" & j + 3).Value = .Range("F" & j + 3).Value - .Range("F" & 1 +
3).Value 'delta x
      .Range("J" & j + 3).Value = .Range("G" & j + 3).Value - .Range("G" & 1 +
3).Value 'delta y
      .Range("K" & j + 3).Value = .Range("H" & j + 3).Value - .Range("H" & 1 +
3).Value 'delta z
      .Range("L" & j + 3).Value = -Sin(.Range("D" & 1 + 3).Value) * .Range("I" & j +
3).Value + Cos(.Range("D" & 1 + 3).Value) * .Range("J" & j + 3).Value 'Easting
      .Range("M" & j + 3).Value = -Cos(.Range("D" & 1 + 3).Value) *
Sin(.Range("C" & 1 + 3).Value) * .Range("I" & j + 3).Value - Sin(.Range("C" & 1 +
3).Value) * Sin(.Range("D" & 1 + 3).Value) * .Range("J" & j + 3).Value +
Cos(.Range("C" & 1 + 3).Value) * .Range("K" & j + 3).Value 'Northing
```

```
    .Range("N" & j + 3).Value = Cos(.Range("C" & 1 + 3).Value) * 
Cos(.Range("D" & 1 + 3).Value) * .Range("I" & j + 3).Value + Cos(.Range("C" & 1 
+ 3).Value) * Sin(.Range("D" & 1 + 3).Value) * .Range("J" & j + 3).Value + 
Sin(.Range("C" & 1 + 3).Value) * .Range("K" & j + 3).Value 'Up
   Next j

   strFormulas2(1) = "=L5-L4"
   strFormulas2(2) = "=M5-M4"
   strFormulas2(3) = "=SQRT(O4^2+P4^2)"

   .Range("O4:Q4").Formula = strFormulas2
   .Range("O4:Q" & LastRow - 1).FillDown
  End With

End Sub


Sub convertm()

  Dim i, j, k, Count As Integer
  Dim carryover, addx, addy, distance As Long

  addx = 0
  addy = 0
  k = 2
  i = Range("P" & Rows.Count).End(xlUp).Row - 3

  With ThisWorkbook.Sheets("Workspace")
  .Range("R4").Value = 0
  .Range("S4").Value = .Range("L" & 4).Value
  .Range("T4").Value = .Range("M" & 4).Value
    For j = 1 To i
      If .Range("R" & j + 3).Value > 0 Then
        If .Range("R" & j + 3).Value + .Range("Q" & j + 3).Value >= 10 Then
          For Count = 1 To Application.WorksheetFunction.Floor((.Range("R" & j + 
3).Value + .Range("Q" & j + 3).Value) / 10, 1)
            If Count = 1 Then
              .Range("S" & k + 3).Value = .Range("S" & k + 3 - 1).Value + .Range("O" 
& j + 3).Value / .Range("Q" & j + 3).Value * (10 - .Range("R" & j + 3).Value) + 
addx
              .Range("T" & k + 3).Value = .Range("T" & k + 3 - 1).Value + .Range("P" 
& j + 3).Value / .Range("Q" & j + 3).Value * (10 - .Range("R" & j + 3).Value) + 
addy
              k = k + 1
            Else
          .Range("S" & k + 3).Value = .Range("S" & k + 3 - 1).Value + .Range("O"
& j + 3).Value / .Range("Q" & j + 3).Value * 10
          .Range("T" & k + 3).Value = .Range("T" & k + 3 - 1).Value + .Range("P"
& j + 3).Value / .Range("Q" & j + 3).Value * 10
          k = k + 1
         End If
        Next Count
        .Range("R" & j + 3 + 1).Value = .Range("Q" & j + 3).Value - (10 -
.Range("R" & j + 3).Value) - (Count - 2) * 10
        addx = .Range("O" & j + 3).Value / .Range("Q" & j + 3).Value * .Range("R"
& j + 3 + 1).Value
        addy = .Range("P" & j + 3).Value / .Range("Q" & j + 3).Value * .Range("R"
& j + 3 + 1).Value
       Else
        .Range("R" & j + 3 + 1).Value = .Range("Q" & j + 3).Value + .Range("R" & j
+ 3).Value
        addx = .Range("O" & j + 3).Value + addx
        addy = .Range("P" & j + 3).Value + addy
       End If
      Else
       If .Range("Q" & j + 3).Value >= 10 Then
        For Count = 1 To Application.WorksheetFunction.Floor((.Range("Q" & j +
3).Value) / 10, 1)
         .Range("S" & k + 3).Value = .Range("S" & k + 3 - 1).Value + .Range("O"
& j + 3).Value / .Range("Q" & j + 3).Value * 10
         .Range("T" & k + 3).Value = .Range("T" & k + 3 - 1).Value + .Range("P" &
j + 3).Value / .Range("Q" & j + 3).Value * 10
         k = k + 1
        Next Count
        .Range("R" & j + 3 + 1).Value = .Range("Q" & j + 3).Value - (Count - 1) * 10
        addx = .Range("O" & j + 3).Value / .Range("Q" & j + 3).Value * .Range("R"
& j + 3 + 1).Value
        addy = .Range("P" & j + 3).Value / .Range("Q" & j + 3).Value * .Range("R"
& j + 3 + 1).Value
       Else
        If .Range("Q" & j + 3).Value < 10 Then
         .Range("R" & j + 3 + 1).Value = .Range("Q" & j + 3).Value
         addx = .Range("O" & j + 3).Value
         addy = .Range("P" & j + 3).Value
        End If
       End If
      End If
    Next j
  End With
End Sub
```

```
Sub Chart()
 Dim myChtObj As ChartObject
 Dim rngChtData As Range
 Dim rngChtXVal As Range
 Dim rngRefData As Range
 Dim rngRefXVal As Range
 Dim RngToCover As Range
 Dim LastRow As Long

 With ThisWorkbook
  Sheets("Workspace").Activate
  LastRow = Range("T" & Rows.Count).End(xlUp).Row

  Set rngChtData = ActiveSheet.Range("T4:T" & LastRow)
  Set rngChtXVal = ActiveSheet.Range("S4:S" & LastRow)
  Set rngRefData = ActiveSheet.Range("AA4:AA5")
  Set rngRefXVal = ActiveSheet.Range("Z4:Z5")
  Sheets("Main").Activate
  Set myChtObj = ActiveSheet.ChartObjects.Add _
   (Left:=250, Width:=375, Top:=75, Height:=225)
  With myChtObj.Chart

  .ChartType = xlXYScatterLines

  ' remove extra series
  Do Until .SeriesCollection.Count = 0
   .SeriesCollection.Delete
  Loop

  ' add series from selected range, column by column
   With .SeriesCollection.NewSeries
    .Values = rngChtData
    .XValues = rngChtXVal
    .Name = "Route Plot"
   End With

   With .SeriesCollection.NewSeries
    .Values = rngRefData
    .XValues = rngRefXVal
    .Name = "1km Reference"
   End With

   Set RngToCover = ActiveSheet.Range("G8:O23")
   myChtObj.Height = RngToCover.Height ' resize
   myChtObj.Width = RngToCover.Width  ' resize
```

```
    myChtObj.Top = RngToCover.Top    ' reposition
    myChtObj.Left = RngToCover.Left   ' reposition
   End With
  End With
End Sub

Sub HeadingXY()

  Dim I As Integer
  Dim strFormulas1(1 To 3), strFormulas2(1 To 1), strFormulas3(1 To 1),
strFormulas4(1 To 1), strFormulas5(1 To 1), strFormulas6(1 To 1) As Variant
  Dim LastRow As Long

  With ThisWorkbook
   .Sheets("Workspace").Select
  End With

  LastRow = Range("T" & Rows.Count).End(xlUp).Row

  With ThisWorkbook.Sheets("Workspace")
    strFormulas1(1) = "=S6-S4"
    strFormulas1(2) = "=T6-T4"
    strFormulas1(3) = "=atan2(U5,V5)/pi()*180"

    .Range("U5:W5").Formula = strFormulas1
    .Range("U5:W" & LastRow - 1).FillDown
    For I = 1 To LastRow - 5
      If .Range("W" & I + 4).Value > 90 And .Range("W" & I + 4).Value < 180 Then
        .Range("X" & I + 4).Value = 360 + (90 - .Range("W" & I + 4).Value)
      Else
        .Range("X" & I + 4).Value = 90 - .Range("W" & I + 4).Value
      End If
    Next I
    strFormulas2(1)            =            "=IF(AND(X6-X5<=540,X6-X5>=180),X6-X5-
360,IF(AND(X6-X5>=-540,X6-X5<=-180),X6-X5+360, X6-X5))"
    .Range("Y5").Formula = strFormulas2
    .Range("Y5:Y" & LastRow - 2).FillDown
    .Range("V" & LastRow + 2).Value = "Mean Heading"
    .Range("W" & LastRow + 2).Value = "Sigma Heading"
    .Range("X" & LastRow + 2).Value = "Mean Change Heading"
    .Range("Y" & LastRow + 2).Value = "Sigma Change Heading"
    .Range("Y" & LastRow + 4).Value = "Road Type"
    strFormulas3(1) = "=AVERAGE(X4:X" & LastRow - 1 & ")"
    strFormulas4(1) = "=STDEVA(X4:X" & LastRow - 1 & ")"
    strFormulas5(1) = "=AVERAGE(Y5:Y" & LastRow - 1 & ")"
    strFormulas6(1) = "=STDEVA(Y5:Y" & LastRow - 1 & ")"
    .Range("V" & LastRow + 3).Formula = strFormulas3
    .Range("W" & LastRow + 3).Formula = strFormulas4
    .Range("X" & LastRow + 3).Formula = strFormulas5
    .Range("Y" & LastRow + 3).Formula = strFormulas6
    If .Range("Y" & LastRow + 3).Value >= 0 And .Range("Y" & LastRow +
3).Value <= 1 Then
      .Range("Y" & LastRow + 5).Value = 1
    ElseIf .Range("Y" & LastRow + 3).Value > 1 And .Range("Y" & LastRow +
3).Value <= 4.5 Then
      .Range("Y" & LastRow + 5).Value = 2
    ElseIf .Range("Y" & LastRow + 3).Value > 4.5 And .Range("Y" & LastRow +
3).Value <= 9.5 Then
      .Range("Y" & LastRow + 5).Value = 3
    ElseIf .Range("Y" & LastRow + 3).Value > 9.5 And .Range("Y" & LastRow +
3).Value <= 14.5 Then
      .Range("Y" & LastRow + 5).Value = 4
    ElseIf .Range("Y" & LastRow + 3).Value > 14.5 And .Range("Y" & LastRow +
3).Value <= 19.5 Then
      .Range("Y" & LastRow + 5).Value = 5
    ElseIf .Range("Y" & LastRow + 3).Value > 19.5 And .Range("Y" & LastRow +
3).Value >= 30.5 Then
      .Range("Y" & LastRow + 5).Value = 6
    Else
      .Range("Y" & LastRow + 5).Value = "ERROR!!"
    End If

    If .Range("V" & LastRow + 3).Value >= 0 And .Range("V" & LastRow +
3).Value <= 90 Then
    .Range("Z4").Value = .Range("S4").Value
    .Range("AA4").Value = .Range("T4").Value - 2000
    .Range("Z5").Value = .Range("S4").Value + 1000
    .Range("AA5").Value = .Range("T4").Value - 2000
    ElseIf .Range("V" & LastRow + 3).Value > 90 And .Range("V" & LastRow +
3).Value <= 180 Then
    .Range("Z4").Value = .Range("S4").Value
    .Range("AA4").Value = .Range("T4").Value + 2000
    .Range("Z5").Value = .Range("S4").Value + 1000
    .Range("AA5").Value = .Range("T4").Value + 2000
    ElseIf .Range("V" & LastRow + 3).Value < 0 And .Range("V" & LastRow +
3).Value >= -90 Then
    .Range("Z4").Value = .Range("S4").Value - 1000
    .Range("AA4").Value = .Range("T4").Value - 2000
    .Range("Z5").Value = .Range("S4").Value
    .Range("AA5").Value = .Range("T4").Value - 2000
```

```
    Else
    .Range("Z4").Value = .Range("S4").Value - 1000
    .Range("AA4").Value = .Range("T4").Value + 2000
    .Range("Z5").Value = .Range("S4").Value
    .Range("AA5").Value = .Range("T4").Value + 2000
    End If
  End With
  Worksheets("Workspace").Range("Y" & LastRow + 5).Copy
Worksheets("Main").Range("F5")
  Worksheets("Main").Range("F5").Interior.ColorIndex = 6
  Worksheets("Main").Range("F5").Font.Name = "Calibri"
  Worksheets("Main").Range("F5").Font.Size = 20
  Worksheets("Main").Range("F5").Font.Bold = True
  Worksheets("Main").Range("F5").HorizontalAlignment = xlCenter
End Sub


Sub RemoveGPX()
With ThisWorkbook

    Application.DisplayAlerts = False
    .Sheets("GPX data").Delete
    Application.DisplayAlerts = True
    '.Sheets("Workspace").Visible = xlSheetVeryHidden
    .Sheets("Main").Select

End With
End Sub
```

LIST OF REFERENCES

[1] K. Osborn. (2017, Apr. 7). Navy praises tomahawk—Cites weapon's future. [Online] Available: http://www.scout.com/military/warrior/story/1695412-navy-praises-tomahawk-cites-weapon-s-future

[2] V. Insinna. (2017, Mar. 13). *Here's how the Air Force is fixing the F-35's moving target problem. Defense News* [Online]. Available: http://www.defensenews.com/articles/heres-how-the-air-force-is-fixing-the-f-35s-moving-target-problem

[3] M. R. Driels, *Weaponeering: Conventional Weapon System Effectiveness*. 2nd ed. Reston, VA: AIAA, Inc., 2013.

[4] S. Andersson and J. Aronsson, "Road shape modeling from digital map data-and implementation of a map supported cruise control," M.S. thesis, Dept. Signals & Systems, Chalmers Univ. of Tech. Göteborg, Sweden, 2009.

[5] E. Donnell, S. Hines, K. Mahoney, R. Porter, and H. McGee. (2009, Sep). Speed concepts: Informational guide. [Online] Available: https://safety.fhwa.dot.gov/speedmgt/ref_mats/fhwasa10001/

[6] N. Aslan, "Maneuverability Estimation of High-Speed Craft," M.S. thesis, Mech. & Aerospace Eng. Dept., NPS, Monterey, CA, 2015.

[7] E.M. Lewandowski. *The Dynamics of Marine Craft*, Washington DC: World Scientific, 2004, pp. 361–396.

[8] S. Denny and E. Hubble, "Prediction of craft turning characteristics," *Marine Technol*., vol. 28, pp. 1–13, 1991.

[9] J. Farrell, and B. Matthew. *The Global Positioning System and Inertial Navigation*, New York, NY: McGraw-Hill, 1999.

[10] M. R. Driels, S. Runyon, and A. Guest. 2016. "Producing TSPI data for moving targets in realistic environment," unpublished.

[11] P. E. Gill, W. Murray, and M. A. Saunders. User's Guide for SNOPT Version 7: Software for Large Scale Nonlinear Programming. [Online]. Available: https://web.stanford.edu/group/SOL/guides/sndoc7.pdf. Accessed Jun. 1, 2017

[12] I. Kaminer, private communication, Jun. 2017.

[13] Icemarine. (n.d.). Bladerunner51. [Online]. Available: www.icemarine.com/models/bladerunner51/. Accessed Jun. 1, 2017

[14] Yachtworld. (n.d.). Boston Whaler 420. [Online]. Available: http://www.yachtworld.com/boats/2016/Boston-Whaler-420-Outrage-3011975/Antigua-%26-Barbuda#.WYzq2FGGNPY. Accessed Jun. 1, 2017

[15] Google Maps. (n.d.). [Online] Available: https://www.google.com/maps/. Accessed Jun. 13, 2017.

[16] S. Sigmundarson. (n.d.). Maps to GPX converter. [Online] Available: https://mapstogpx.com. Accessed Jun. 13, 2017.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California